#123 January 1987

2.95 (3.95 CANADA)

# Dr. Dobb's Journal of Coffware Tools

FOR THE PROFESSIONAL PROGRAMMER

#### ANNUAL 68K ISSUE

68K Mini Forth

OS-9 Operating
System

Mac and Amiga Interface Programming

#### Languages:

Forth Names
Proper PROLOG
Memory Management in C
BASIC Rebirth
68K Assembly Project

The Bandwidth Bottleneck



# Btrieve.

### The Programmer's Choice.

hen you're serious about application development, there's just one choice for file management: Btrieve. With what Computer Language calls "near mainframe functionality", Btrieve sets the file management standard for PC applications. With Btrieve loaded in your PC, your programs can use simple subroutine calls to retrieve, store and update records.

B-tree based for high performance. Performance is all-important, especially as your database grows. That's why Btrieve implements the b-tree file structure—the fastest, most efficient method of accessing data.

Interfaces to C, BASIC, Pascal, COBOL. Don't waste time

Help is

just a phone

Btrieve users receive 30 days

of unlimited phone support at no charge. This

"Direct Connect" policy is renewable for a full

bulletin board for technical tips, seven days a week.

year at low cost. And try SoftCraft's free

call away. Need technical support? You've got it!

programming in awkward fourth generation languages! With Btrieve, simply use the languages you know best—and write applications the right way. Over 15 language interfaces available.

LANs, including IBM PC Network and Novell Advanced Netware. Btrieve is also available for Xenix and multitasking operating systems such as MultiLink Advanced, Microsoft Windows and IBM Topview.

Built-in security

features. Lock up sensitive data with Btrieve's password protection and unique data encryption scheme—especially useful in local area networks.



Thorough documentation, easy implementation. Getting started with Btrieve is easy: the manual is packed with examples of every Btrieve function in BASIC, Pascal, COBOL and C.

Multi-user versions for

LANs and Xenix. When your

applications need to network, count on

Btrieve. A single version runs on all DOS 3

Database queries, report writing. Add Xtrieve™ to your Btrieve applications for a fully-relational DBMS. Xtrieve's menu-driven interface gives your users the on-line query capabilities they need—without programming. Add

our report writer option to produce custom reports and forms.



*No royalties.* Need we say more?



Fault tolerant. Btrieve insures against database disasters. Two levels of fault tolerance guarantee data integrity during accidents or power failures—even if lightning strikes. No extra programming required.





P.O. Box 9802 #917

Austin, Texas 78766 (512) 346-8380 Telex 358 200

Suggested retail prices: Btrieve, \$245; multi-user Btrieve, \$595; Xtrieve, \$245; multi-user Xtrieve, \$595 (for report generation, add \$145 for single-user and \$345 for multi-user). Available from SoftCraft and selected distributors. Requires PC-DOS or MS-DOS 2.X, 3.X, Xenix. Btrieve is a registered trademark and Xtrieve is a trademark of SoftCraft Inc. 1From Computer Language, November 1985.

# TRANSFORM YOUR PC INTO A LISP MACHINE.



Not long ago, a specialized LISP machine was your only choice for serious AI development and delivery.

But now you can get LISP machine performance out of that ordinary IBM PC\* sitting on your desk. With the Gold Hill 386 LISP System.

You simply plug in the System's HummingBoard™— unique 386-based hardware designed specifically for the LISP environment. Then you add the System's Golden Common LISP 386 Developer software.

That's all you need to transform your PC into a LISP machine. You can develop and deliver AI applications on your PC. And create your own expert systems. You'll get the kind of LISP performance you thought was only possible in a system costing ten times as much.

And if you don't need the whole system, Gold Hill can still offer you AI solutions. You can get GCLISP 286 Developer software for your PC.\* And GCLISP 386 Developer software will be available for leading manufacturers' 386-based PCs.

Tomorrow's AI development tools for PCs are available today from Gold Hill. We'll prove it to you—just call to get the latest Gabriel Performance Benchmarks. You'll be amazed to learn what your PC is capable of. Call toll-free:

#### Gold Hill. The expert in AI on PCs.

\*The Gold Hill 386 LISP System requires an IBM PC XT, AT or compatible. GCLISP 286 Developer Software requires an IBM PC AT or compatible.
© 1986 Gold Hill Computers, Inc. Gold Hill, Gold Hill 386 LISP System, Golden Common LISP, GCLISP, and Developer are trademarks of Gold Hill Computers, Inc. IBM PC, XT and AT are registered trademarks of International Business Machines Corp. HummingBoard is a trademark of A.I. Architects, Inc.

Circle no. 291 on reader service card

#### 1-800-242-LISP

In Mass.: (617) 492-2071 Gold Hill Computers, Inc. 163 Harvard St., Cambridge, MA 02139

GOLDHILL

# Instant Replay

Has It Dawned On You Yet? Prototype-->Program-->Tutorial

#### ) Instant Replay - \$ 89.95

Demo, Tutorial, and Prototype Generator.
Create Replays of actual programs, or vapor ware only replays. Includes Screen Genie Screen Editor, Word Genie Text Editor, and a Keystroke Editor. Insert Pop-Ups, Prompts, Prototypes, and Users into the demo or tutorial.

#### ) Assembler Genie \$ 59.95

Assembly Language source code generator for including Screen Genie prototype screens into your actual programs. Supports almost every language.

#### ) Plus Series - \$ 59.95

Assembly Language Programming Tools for Turbo Pascal, MS Pascal, or C. Make your prototypes into real programs.

#### NoBlink Accelerator - \$ 49.95

Resident Cursor Enhancer. Choose your own cursor shape, color(non-blinking), or speed in any program. Uses only 6K of memory. Great time time saver. Why mouse around?

50 day satisfaction money back guarantee on all software. Not Copy Protected.

o order, or to obtain more information call or write ostradamus Inc. (801) 487-9662
91 South Valley Street (suite 252)

t Lake City, Utah 84109

NOSTRADAMUS Software Sourcery

Circle no. 251 on reader service card.

Amex, Master, Cod, PO, Check
M PC, XT, AT and True Compatibles.

e US pays postage.

#### CONTENTS

OF INTEREST:

10

14

130

152

New products out there

151

ADVERTISER INDEX:

Where to find those ads

The 68K story	680xx PROGRAMMING: 680x They Going? by Nick Turner An overview of the 680xx familiand probable future. 680xx PROGRAMMING: A Minby G. Yates Fletcher Yates tells us about the "no frill the 68000 that he designed to temore naturally understood as a language. 680xx PROGRAMMING: The 6by Brian Capouch A look at the modular, multipress	ly of chips: past, present,  ni Forth for the 68000  22  23  25" Forth-like interpreter for est the theory that Forth is a program than as a  OS-9 Operating System  36					
Mac and Amiga assembly ▶	operating system growing in poprogrammers.  680xx PROGRAMMING: Macin	opularity among 680xx					
language	Gadgets by Jan L. Harrington Comparing user interface programming on the Macin and Amiga, Jan provides details about the operating sy support produced on both machines for user interface features such as menus, buttons and windows. PROCESSORS: Series 32000 Cross Assembler by Richard Rodman The listing (in human-readable form) for Richard's artithat was published in December.						
Managing your	COL	JMNS					
memory >	by Allen Holub Allen presents some memory memory and plus an explanation of C memory memory and the company of C memory memory memory memory memory and the company of C memory m	104 nanagement techniques ry organization for					
Forth names	beginning C programmers.  STRUCTURED PROGRAMMING by Michael Ham Michael discusses the naming of names in Forth.  THE RIGHT TO ASSEMBLE by Nick Turner Nick launches a project to design a versatile, easy-to-use, interpreted language to be written in 680xx assembly lar						
The bandwidth 3	FORUM	PROGRAMMER'S SERVICES					
bottleneck <b>Z</b>	EDITORIAL 6 by Michael Swaine RUNNING LIGHT 8	DR. DOBB'S CATALOG: 117 DDJ books and software OF INTEREST: 140					

by Nick Turner

**ARCHIVES** 

VIEWPOINT

by Dick Butrick **DDJ ON LINE** 

**SWAINE'S FLAMES** 

by Michael Swaine

**LETTERS** 

by you



**About the Cover** 

Motorola has just forged the 68030. Is it as hot as it seems?

#### This Issue

In the beginning there were the 8080 and the 6502-programmers chose their weapons and the battle lines were drawn. A few years later, Motorola gave the "sixers" more power when it introduced its 680xx line of chips. Today there is a wide range of powerful 680xx machines-and some very interesting rumors about the future. This month we survey the 680xx family and examine a modular, multitasking operating system, a 68K Forthlike interpreter, and the challenges of creating Amiga- and Mac-like user interfaces.

#### **Next Issue**

The choice of a text editor is based on many highly subjective considerations as well as some "hard" pragmatic requirements. In February, we'll present an overview of the various elements involved in that choice and let you hear what some programmers have to say about their favorite and least favorite editors.

-bandwidth topic



BASIC

Proper PROLOG

The rebirth of

YOUR
COMPUTER LANGUAGE
IS QUIETLY
BREEDING REAL BATS
IN YOUR
BELFRY.

# LANGUAGES THAT ARE CAUSING THE BIGGEST PROGRAMMING BACKLOG IN HISTORY ARE ALSO

Whether it's BASIC, COBOL, Pascal, "C", or a data base manager, you're being held back.

Held back because the language has frustrating limitations, and the programming environment isn't intuitive enough to keep track of what you're working on.

In the real world, there's pressure to do more impressive work, in less time, and for more clients.

We've been given some incredibly powerful hardware in recent times, but the languages aren't a whole lot better than they were 20 years ago. So, whatever language you have chosen, by now you feel it's out to

get you — because it is.

Sure, no language is perfect, but you have to wonder, "Am I getting

all I deserve?"

And, like money, you'll never

have enough.

Pretty dismal, huh? We thought so, too. So we did something about it. We call it CLARION™

You'll call it "incredible." Distributed on 7 diskettes, CLARION consists of over 200,000 lines of code, taking 3+ years to hone to "world-class" performance.

With CLARION you can write, compile, run and debug complex applications in a New York afternoon.

Even if you're in Savannah. It gives you the power and speed to create screens, windows and reports of such richness and clarity you would never attempt them with any other language.

Because you would have to

write the code.

With CLARION you simply design the screens using our SCREENER utility and then CLARION writes the source code AND compiles it for you in seconds.

Likewise, you can use REPORTER to create reports.

Remember, only CLARION can recompile and display a screen or report layout for modification. And with no time wasted.

All the power and facilities you need to write great programs, faster than you ever dreamed of.

Programs that are easy to use. Programs that are a pleasure to

And to you that means true satisfaction.

You've coveted those nifty pop-up help windows some major applications feature. But you can't afford the time and energy it takes to write them into your programs.

That's the way it used to be.

So we fixed that, too. CLARION'S HELPER is an interactive utility that let's you design the most effective pop-up help screens that you can imagine. And they're "context sensitive," meaning you can have help for every field in your application.

Unlike the other micro languages, CLARION provides declarations, procedures, and functions to process

dates, strings, screens, reports, indexed files. DOS files and

Imagine making source program changes with the CLARION EDI-TOR. A single keystroke terminates the EDITOR, loads the COM-PILER, compiles the program, loads the PROCESSOR and executes the program. It's that easy!

Our data management capabilities are phenomenal. CLARION files permit any number of composite keys which are updated dynami-

cally.

A file may have as many keys as it needs. Each key may be composed of any fields in any order. And key files are updated when-ever the value of the key changes. Like SCREENER and RE-PORTER, CLARION'S FILER utility

also has a piece of the CLARION COMPILER. To create a new file, you name the Source Module. Then you name the Statement Label of a file structure within it.

FILER will also automatically rebuild existing files to match a changed file structure. It creates a new record for every existing record, copying the existing fields and initializing new ones.

Sounds pretty complicated, huh?
Not with CLARION's documentation and on-line help screens. If you are currently competent in BASIC, Pascal or "C" you can be writing CLARION applications in a day. In two days you won't believe the eloquence of your CLARION programs. Okay, now for the best part of

all. You can say it in CLARION for \$295.00—plus shipping and handling. All you need is an IBM® PC, XT, AT or true compatible, with 320 KB of memory, a hard disk drive, and a parallel port. And we'll allow a full 30 day evaluation

period. If you're not satisfied with CLARION, simply return it in its original condition for a full refund.

If you're not quite ready to take advantage of this no-risk opportunity, ask for our detailed 16 page color brochure. It vividly illustrates the elegance of CLARION. Consider it a preview of programming in the fast lane.

Either way, the 800 call's a freebie.



A4ST/8

**SAY IT IN** 







150 EAST SAMPLE ROAD POMPANO BEACH, FLORIDA 33064 305/785-4555 BARRINGTON SYSTEMS, INC.

IBM is a registered trademark of International Business Machines Corporation. CLARION™ is a trademark of Barrington Systems, Inc. ©1986 Barrington Systems Circle no. 115 on reader service card.

ere's some of what we have planned for DDJ's twelfth year.

In a series of short reports, contributing editor Namir Shammas will examine the state of the BASIC language in 1987. Yes, BA-SIC. DDJ is hardly a beginner's magazine, but

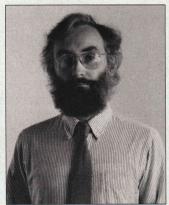
if the beginner's language has grown up to long pants, as some claim, we should all know about it.

Then again, each of us is a novwice in some area. Realizing this, DDJ will use the icon at the beginning of this paragraph to flag certain features as entry points: items such as Allen Holub's Flotsam and Jetsam in this issue, from which the less experienced programmer can learn useful techniques or gain familiarity with more technical subjects.

There are a lot of ways to write about artificial intelligence, nearly all bad. Our new column on alternative programming paradigms (it starts next month) will avoid them all as it examines such topics as knowledgebased programming, logic programming, and object-oriented programming from an experienced software designer's perspective. Contributing editor Ernie Tello writes, lectures, and consults in this area and promises to take us beyond the fields we know.

And we're going to attack the bandwidth problem.

DDJ was born to shoehorn BASIC into the hypomnemonic personal computers of 1976. In a sense, the Cain/Hendrix versions of Small-C published in DDJ over the succeeding years addressed this same problem of cramming programming power into micro memory. You could say that's been the charter of the magazine, and on one level it will continue to be our focus. But nobody today needs another Tiny BASIC or Small-C. Developments like the Intel 80386 proces-



sor lift the lid of a different box of programming problems. One is the efficient transmission of information over limited channels.

Bandwidth is already an issue in graphics output: Microsoft Windows never made sense on 8088 machines and the PARC

interface overwhelmed the original 128K Mac. Adequate memory and processor power make a big difference, but they may ultimately just move the bottleneck to the IC level.

Bandwidth becomes more of an issue in mass storage as storage becomes more massive. Once we learn what to do with CD-ROMs, retrieving information efficiently from them will require more than increased speed of transmission.

The potential bandwidth crunch in telecommunications and remote database access is obvious, but when LANs start proliferating, so will inhouse bandwidth competition.

Approaches to the bandwidth crunch can range from clever datacompression algorithms to systems that form hypotheses about incoming data and acquire just enough data to confirm or reject the hypotheses. As access to information becomes more technically problematic, it will also take on sociopolitical dimensions; for example, public libraries are becoming measurably less public as they subscribe to commercial informationprovider services and pass the costs on to their patrons. We'll delve into the technology for dealing with the bandwidth crunch while trying to see its potential social consequences.

Bandwidth-related items will be flagged with the icon at the end of this line.

> Milail Michael Swaine editor-in-chief

#### Dr. Dobb's Journal of **Software Tools**

Editor-in-Chief Michael Swaine

Editor Nick Turner **Managing Editor** Vince Leone

**Assistant Editors** Sara Noah Ruddy

Levi Thomas

Technical Editor Contributing Editors Ray Duncan

Michael Ham Bela Lubkin

Namir Shammas Ernest R. Tello

Copy Editor Rhoda Simmons

Production

Production Manager Bob Wynne Michael Hollister Art Director

Assoc. Art Director Joe Sikoryak Typesetter Jean Aring

Cover Artist Michael Carr

Circulation

**Circulation Director** Maureen Kaminski Newsstand Sales Mgr. Stephanie Barber Jane Sharninghouse Book Marketing Mgr. Circulation Coordinator Kathleen Shay

Administration

Finance Director Kate Wheat **Business Manager** Betty Trickett

Accounts Payable Supv. Mayda Lopez-Quintana

Accounts Receivable Mgr. Laura Di Lazzaro

**Advertising Director** Bobert Horton (415) 366-3600

**Account Managers** 

Michele Beaty (317) 875-8093

(415) 366-3600 Lisa Boudreau

Gary George (404) 897-1923 Michael Wiener (415) 366-3600

Cynthia Zuck (718) 499-9333 Promotions/Srycs.Mgr. Anna Kittleson Advertising Coordinator Charles Shively

#### M&T Publishing Inc.

Chairman of the Board Otmar Weber Director C.F. von Quadt President and Publisher Laird Foshay

Dr. Dobb's Journal of Software Tools (USPS 307690) is published monthly by M&T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600. Second-class postage paid at Redwood City and at additional entry points.

Article Submissions: Send manuscripts and disk (with article and listings) to the Editor.

DDJ on CompuServe: Type GO DDJ

Address Correction Requested: Postmaster: Send Form 3579 to Dr. Dobb's Journal, P.O. Box 27809, San Diego, CA 92128. ISSN 0888-3076

Customer Service: For subscription problems call: outside CA (800) 321-3333; in CA (619) 485-9623 or 566-6947. For book/software order problems call (415) 366-

Subscriptions: \$29.97 per year; \$56.97 for 2 years. Canada and Mexico add \$27 per year airmail or \$10 per year surface. All other countries add \$45 per year airmail or \$28 per year surface. Foreign subscriptions must be prepaid in U.S. funds drawn on a U.S. bank. For foreign subscriptions, TELEX: 752-351.

Foreign Newsstand Distributor: Worldwide Media Service Inc., 386 Park Ave. South, New York, NY 10016; (212) 686-1520 TELEX: 620430 (WUI).

Entire contents copyright © 1987 by M&T Publishing, Inc., unless otherwise noted on specific articles. All rights reserved.

#### People's Computer Company

Dr. Dobb's Journal of Software Tools is published by M&T Publishing Inc. under license from People's Computer Company, 2682 Bishop Dr., Suite 107, San Ramon, CA 94583, a nonprofit corporation.



#### The C for Microcomputers

PC-DOS, MS-DOS, CP/M-86, Macintosh, Amiga, Apple II, CP/M-80, Radio Shack, Commodore, XENIX, ROM, and Cross Development systems

#### MS-DOS, PC-DOS, CP/M-86, XENIX, 8086/80x86 ROM

#### Manx Aztec C86

"A compiler that has many strengths... quite valuable for serious work"

Computer Language review, February 1985

Great Code: Manx Aztec C86 generates fast executing compact code. The benchmark results below are from a study conducted by Manx. The Dhrystone benchmark (CACM 10/84 27:10 p1018) measures performance for a systems software instruction mix. The results are without register variables. With register variables, Manx, Microsoft, and Mark Williams run proportionately faster, Lattice and Computer Innovations show no improvement.

	Execution Time	Code Size	Compile/ Link Time
Dhrystone Benchmark			
Manx Aztec C86 3.3	34 secs	5,760	93 secs
Microsoft C 3.0	34 secs	7,146	119 secs
Optimized C86 2.20J	53 secs	11,009	172 secs
Mark Williams 2.0	56 secs	12,980	113 secs
Lattice 2.14	89 secs	20,404	117 secs

Great Features: Manx Aztec C86 is bundled with a powerful array of well documented productivity tools, library routines and features.

Optimized C compiler
AS86 Macro Assembler
80186/80286 Support
8087/80287 Sensing Lib
Extensive UNIX Library
Large Memory Model
Z (vi) Source Editor -c
ROM Support Package -c
Library Source Code -c
MAKE, DIFF, and GREP -c
One year of updates -c

Symbolic Debugger LN86 Overlay Linker Librarian Profiler DOS, Screen, & Graphics Lib Intel Object Option CP/M-86 Library -c INTEL HEX Utility -c Mixed memory models -c Source Debugger -c CP/M-86 Library -c

Manx offers two commercial development systems, Aztec C86-c and Aztec C86-d. Items marked -c are special features of the Aztec C86-c system.

Aztec C86-c Commercial System	\$499
Aztec C86-d Developer's System	\$299
Aztec C86-p Personal System	\$199
Aztec C86-a Apprentice System	\$49

All systems are upgradable by paying the difference in price plus \$10.

Third Party Software: There are a number of high quality support packages for Manx Aztec C86 for screen management, graphics, database management, and software development.

C-tree \$395 Greenleaf \$185
PHACT \$250 PC-lint \$98
HALO \$250 Amber Windows \$59
PRE-C \$395 Windows for C \$195
WindScreen \$149 FirsTime \$295
SunScreen \$99 C Util Lib \$185
PANEL \$295 Plink-86 \$395

#### MACINTOSH, AMIGA, XENIX, CP/M-68K, 68k ROM

#### Manx Aztec C68k

"Library handling is very flexible ... documentation is excellent ... the shell a pleasure to work in ... blows away the competition for pure compile speed ... an excellent effort."

Computer Language review, April 1985

Aztec C68k is the most widely used commercial C compiler for the Macintosh. Its quality, performance, and completeness place Manx Aztec C68k in a position beyond comparison. It is available in several upgradable versions.

Optimized C Macro Assembler Overlay Linker Resource Compiler Debuggers Librarian Source Editor MacRam Disk -c Library Source -c Creates Clickable Applications Mouse Enhanced SHELL Easy Access to Mac Toolbox UNIX Library Functions Terminal Emulator (Source) Clear Detailed Documentation C-Stuff Library UniTools (vi,make,diff,grep) -c

One Year of Updates -c

Items marked -c are available only in the Manx Aztec C86-c system. Other features are in both the Aztec C86-d and Aztec C86-c systems.

Aztec C68k-c Commercial System	\$499
Aztec C68d-d Developer's System	\$299
Aztec C68k-p Personal System	\$199
C-tree database (source)	\$399
AMIGA, CP/M-68k, 68k UNIX	call

#### Apple II, Commodore, 65xx, 65C02 ROM

#### Manx Aztec C65

"The AZTEC C system is one of the finest software packages I have seen"

NIBBLE review, July 1984

A vast amount of business, consumer, and educational software is implemented in Manx Aztec C65. The quality and comprehensiveness of this system is competitive with 16 bit C systems. The system includes a full optimized C compiler, 6502 assembler, linkage editor, UNIX library, screen and graphics libraries, shell, and much more. The Apple II version runs under DOS 3.3, and ProDOS, Cross versions are available.

The Aztec C65-c/128 Commodore system runs under the C128 CP/M environment and generates programs for the C64, C128, and CP/M environments. Call for prices and availability of Apprentice, Personal and Developer versions for the Commodore 64 and 128 machines.

Aztec C65-c ProDOS & DOS 3.3 \$399
Aztec C65-d Apple DOS 3.3 \$199
Aztec C65-p Apple Personal system \$99
Aztec C65-a for learning C \$49
Aztec C65-c/128 C64, C128, CP/M \$399

#### Distribution of Manx Aztec C

In the USA, Manx Software Systems is the sole and exclusive distributor of Aztec C. Any telephone or mail order sales other than through Manx are unauthorized.

#### **Manx Cross Development Systems**

Cross developed programs are edited, compiled, assembled, and linked on one machine (the HOST) and transferred to another machine (the TARGET) for execution. This method is useful where the target machine is slower or more limited than the HOST, Manx cross compilers are used heavily to develop software for business, consumer, scientific, industrial, research, and educational applications.

HOSTS: VAX UNIX (\$3000), PDP-11 UNIX (\$2000), MS-DOS (\$750), CP/M (\$750), MACINTOSH (\$750), CP/M-68k (\$750), XENIX (\$750).

TARGETS: MS-DOS, CP/M-86, Macintosh, CP/M-68k, CP/M-80, TRS-80 3 & 4, Apple II, Commodore C64, 8086/80x86 ROM, 68xxx ROM, 8080/8085/Z80 ROM, 65xx ROM.

The first TARGET is included in the price of the HOST system. Additional TARGETS are \$300 to \$500 (non VAX) or \$1000 (VAX).

Call Manx for information on cross development to the 68000, 65816, Amiga, C128, CP/M-68K, VRTX, and others.

#### CP/M, Radio Shack, 8080/8085/Z80 ROM

#### Manx Aztec CII

"I've had a lot of experience with different C compilers, but the Aztec C80 Compiler and Professional Development System is the best I've seen."

80-Micro, December, 1984, John B. Harrell III

Aztec C II-c (CP/M & ROM)	\$349
Aztec C II-d (CP/M)	\$199
C-tree database (source)	\$399
Aztec C80-c (TRS-80 3 & 4)	\$299
Aztec C80-d (TRS-80 3 & 4)	\$199

#### How To Become an Aztec C User

To become an Aztec C user call 1-800-221-0440 or call 1-800-832-9273 (800 TEC WARE). In NJ or outside the USA call 201-530-7997. Orders can also be telexed to 4995812.

Payment can be by check, COD, American Express, VISA, Master Card, or Net 30 to qualified customers.

Orders can also be mailed to Manx Software Systems, Box 55, Shrewsbury, NJ 07701.

#### **How To Get More Information**

To get more information on Manx Aztec C and related products, call 1-800-221-0440, or 201-530-7997, or write to Manx Software Systems.

#### 30 Day Guarantee

Any Manx Aztec C development system can be returned within 30 days. for a refund if it fails to meet your needs. The only restrictions are that the original purchase must be directly from Manx, shipped within the USA, and the package must be in resalable condition. Returned items must be received by Manx within 30 days. A small restocking fee may be required.

#### Discounts

There are special discounts available to professors, students, and consultants. A discount is also available on a "trade in" basis for users of competing systems. Call for information.

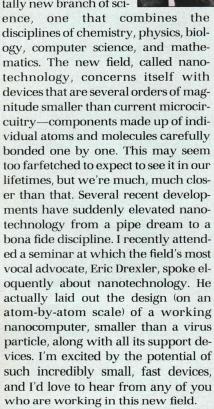


To order or for information call:

800-221-0440

#### RUNNING LIGHT

hat is the logical end point of the current trend toward smaller and smaller chip components? How soon will it be reached, and how reliable will the components be at that level? These questions are being addressed by a totally new branch of sci-



We've received excellent responses to our October 1986 issue, which focused on the 80386 and its family. This month we focus on the other side, the sixers. (If you're a 680xx programmer, you're probably a sixer.) We start with an overview of the 68000 line—where it has been, where it is now, and where it might be going. As more information becomes available on the 68040 (and beyond), we'll keep you up to date.

Yates Fletcher has written a Forth-



like interpreter for the 68000 that he calls FLINT. His article in this issue describes it in detail and offers some interesting insight into what makes threaded interpreters so efficient. Also in this issue, Brian Capouch takes a look at the OS-9 operating system from

Microware. OS-9 shows promise as a standard for 68K development work, and Brian explains why. Jan Harrington's article about Amiga gadgets and Macintosh buttons is one of the clearest comparisons I've seen of the different programming styles required on the two machines.

This month I begin what I hope will be a series of essays on the design of an interpreted language for the 68000. I'm interested in your comments and criticism.

Last July, in our annual Forth issue, we published an article about a Forth-driven robot that dives into the ocean, records oceanic data, and then pops up to send the data home via satellite. This July we'd like to focus on embedded systems of that sort—programs that reside inside autonomous devices. If you're working on such a device, we'd like to hear from you.

Bela Lubkin has joined Levi Thomas and Ray Duncan as a sysop on our CompuServe SIG (DDJFORUM) and has been stirring things up quite nicely. You will doubtless see his name on many a message in DDJ On Line.

Finally, I'd like to thank Jerry Houston, Roger Dunn, John Berry, Charles Marslett, and Wayne Vucenic for their help with Table 2 on page 18.

Nick Turner editor

#### **ARCHIVES**

#### The 68000 et Famille

[A small boy, Oliver Wendall Jones, is sitting on Santa's lap reciting his Christmas wish list.]

Oliver: "That's the 32-bit MC68000 microprocessor ... not to be confused with. ..." Santa: "The 16-bit 8088. Total garbage, El Stinko."—Bloom County (comic strip), Berke Breathed, December 12, 1985.

"Tom Pittman, the implementor of the \$5/copy version of Tiny BASIC for the 6800 and 650x is now offering an experimenter's kit for those folks who want to modify and extend his Tiny BASIC. The kit includes an assembled source listing for the IL (Intermediate Language), an IL assembler written in Tiny BASIC, a detailed description of the virtual machine implemented by the IL, instructions for incorporating a new IL into the Tiny BASIC system, and finally some practical hints about debugging and extending the system. The cost is \$10 from Itty Bitty Computers."—DDJ, March 1977.

#### **Flak from Our Readers**

"A language design frequency of almost one Tiny BASIC version per month is certainly impressive. Programmers ought to have realized that God invented many different laguages in Babylon not to enjoy but to punish mankind, and, after all, he never intended implementing all of them for the 8080. So why not write—and, if necessary, publish—programs in an informal, ad hoc language fitting to the problem, not the computer?"—Thomas Alexander Matzner, letter to the editor, DDJ, March 1977.

"[You write 'em; we'll publish (maybe).]"—editorial reply to the above, DDJ, March 1977.

#### Ten Years Ago in DDJ

"Why bother with a multi-tasking operating system on a 'personal' computer? Let's daydream for a moment. Wouldn't it be nice to be able to start a lengthy listing on our hardcopy device; while that was running, start an assembly of a large program; and then go about editing the source for another program from our softcopy terminal? That's EXACTLY what you can do with a multi-tasking system."—Jim Warren,DDJ, January 1977.

"I would like to particularly applaud Dick [Wilcox]'s position regarding low-cost distribution of software for not-for-profit use. He is recognizing and adjusting to the realities of the new world of personal computing in a manner that I feel is fair and reasonable for everyone concerned."—

Jim Warren, DDJ, January 1977.

DR. DOBB'S JOURNAL of COMPUTER Calisthenics & Orthodontia

### Large model C compiler performs like big-name brands includes free C tutorial

"As good as or better than most of the heavyweights..." DR. DOBB'S, August 1986

The DATALIGHT C COMPILER is a full-implementation of the C language as defined in The C Programming Language by Kernighan and Ritchie. Supporting five memory models, DATALIGHT C has very fast compile, link, and execution times with a minimum of memory required. Our special introductory price is only \$99 for the DEVELOPER'S KIT.

#### **Optimize Your Code Generation**

Now you can produce highly optimized code in the standard Intel object module format. The optimizations performed include common subexpression elimination, branch optimizations, constant folding, strength reductions, dead-code eliminations, and switch table compaction.

#### **Five Memory Models Supported**

DATALIGHT C provides five memory models so you can use the model that best suits your application.

#### **Memory Models**

Model	Code	Data
Compact	64k	total code & data
Small	64k	64k
Program	1M	64k
Data	64k	1M
Large	1M	1M

#### Compiling, One step...

Now with the one step DLC program you can compile, link, and create a .COM file. Also, one or more files can be compiled and linked using DLC.

#### **Complete Library Includes Source Code**

The UNIX compatible library includes complete source code. Experienced programmers can use the source code to configure and rebuild the library to suit the application.

#### Concise Documentation Included

The DATALIGHT C COMPILER and DEVELOPER'S KIT include a concise, to the point, programmer's manual. The 210-page manual is contained in an IBM-style three-ring binder which includes nine chapters, appendices, index, and easy to follow examples.



BOX 82441 KENMORE, WA 98028 (206) 367-1803

#### **30-DAY MONEY-BACK GUARANTEE**

Try our DEVELOPER'S KIT for 30 days, and if you do not find it to be equal to, or better than, any compiler you have, or are using, then you may return it for a full refund. Also, if you return the compiler within 30 days, you may keep the C tutorial, a \$39 value, for trying our DEVELOPER'S

#### **Introductory Prices**

DATALIGHT C without source \$60 (compact & small memory model support)

DEVELOPER'S KIT with source \$99 (five memory models supported)

Not Copy Protected

#### ORDER TOLL-FREE TODAY!



1-(800) 221-6630

VISA

#### \*SPECIAL OFFER!

ORDER the DEVELOPER'S KIT by January 31, 1987 and receive free....



"Combines print and software technology to create the most integrated new type of training system I have ever seen.

#### ADAM GREEN, INFO WORLD January 27, 1986

"C:A Programming Workshop" is a complete Clanguage learning package with an integrated compiler and screen editor. Learn the C language with tutorials, quizzes, and program

Add \$5 for shipping in US/\$15 outside US COD (add \$2.50)

\*Limited offer available exclusively to readers who purchase directly from DATALIGHT.

#### Magazine Reviewers Shocked by DATALIGHT'S PERFORMANCE...

"Reviewing this compiler was quite a surprise for us. For such a low price, we were expecting a "lightweight" compiler. What we got was a package that is as good as or better than most of the "heavyweights." DATALIGHT C implements a complete C language. It also compiles quickly, doesn't take up much disk space, and looks impressive in the benchmarks."

DR. DOBB'S, August 1986

"This is a sharp compiler! ... what is impressive is that DATALIGHT not only stole the compile time show completely, but had the fastest Fibonacci executable time and had excellent object file sizes to boot!"

> Chris Skelly, COMPUTER LANGUAGE February 1986

#### **DEVELOPER'S KIT (VERSION 2.12)**

- Full UNIX System 5 C language plus ANSI extensions.
- · Fast/tight code via powerful optimizations including common sub-expression elimination.
- · DLC one step compile/link program.
- Multiple memory model support.
- · UNIX compatible library with PC functions.
- Compatible with DOS linker and assembler.
- · Third-party library support.
- · Automatic generation of .COM files.
- · Supports DOS pathnames, wild cards, and Input/ Output redirection.
- Compatible with Lattice C version 2.x.
- Interrupt handling in C.
- · Debugger support.
- · ROMable code support/start-up source.

#### **MAKE Maintenance Utility**

- · Macro definition support.
- MS-DOS internal commands.
- · Inference rule support.
- · TOUCH date manager.

#### **Tools in Source Code**

- cat-UNIX style "type"
- diff—Text file differences
- fgrep-Fast text search
- pr—Page printer
  pwd—Print working directory
- wc-Word count



#### LETTERS



#### Software Gap

Dear DDJ,

After reading Nick Turner's Running Light column in the August issue concerning the perceived "software gap," I felt as though I had to tell you how I see it.

I hold a Bachelor of Science degree in computer science from the University of Maryland. I learned all about the "right" way to design and code programs, including everything you say programmers don't care about anymore. From my own experience, I find this to have been a waste of time and tuition.

The biggest problem I have had in my "career" has been convincing

some of the "data processing managers" how a program should be constructed. Every time I try to write some form of documentation, I am told "not to waste time on such useless paperwork." I wish I could say this only happened at one or two companies, but I have been employed at four different companies over the past five years, and every one of them has been the same. I am beginning to think the only way I'll ever get to be what you call a "professional" programmer is to start my own software company.

At times I wonder if the problem with the management structure stems from the fact that most of the people promoted have business backgrounds. Not one of my bosses has ever had anything other than a business degree, and not one knows the first

thing about a programming project. My current boss only understands straight-line coding and sequential list processing (that is, no doubly linked lists, no sparse matrices, no queues, and so on)—nice way to run a systems software development group that still uses a one-way line editor.

All I want is the chance to use what I learned in school and to be able to do a programming project correctly. It would be such a treat.

Name withheld by request

Dear DDJ,

I am writing in response to Nick Turner's August Running Light about sloppy programmers. I am a programmer/engineer, and I see a lot of sloppy programming. In fact, I do some sloppy programming myself. I think I might have a clue as to what is going on.

Turner mentions the programmers who can "write entire operating system kernels... in one pass, in assembly code, that run perfectly the first time..." Well, I think they are the exception to the rule. I do not, however, doubt the premise that we can all do it, it just takes the rest of us

a little more time and concentration.

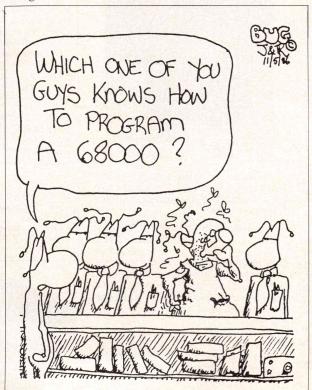
So why don't we take the time, concentrate, and code better? I propose that the reason why we don't is the reward system that most of us work within. My manager seems to be less concerned that a project works the first time than he does that it is done. He wants to "see something." Granted, I work for a defense contractor and my manager wants to record milestones, not finish projects. But I think this attitude is more prevalent than most of us think.

Why, then, does this attitude and reward system not affect the hot performers? I think it is because they, at some level, do not respond to the same reward system as the majority. Every organization has at least one programmer who walks to the beat of a different drummer. This is not to say that all individuals who fit this description are great programmers, but most of the great programmers I know (or know of) are of this type. On the other hand, this tends to make them more difficult to manage, partially because they do not respond to the reward system that reflects the views of management.

So what can be done? Well, we

probably won't change the managers or their ideas of how things should be done. From my own experience, I find that if they want to see something I have two choices. First, I can slop it together and debug it later (when I have less time). Second, I can "stub" it off, perhaps only coding the user interface or some visual part of the program. This lets the manager see something, but the code he sees is good code. Later, I go back and, instead of debugging, I write (for the first time) the code that I stubbed off in the first place. This seems like the way to go, but it is sometimes difficult to determine when to stop coding and start stubbing.

So I haven't really offered a solution. Just some ideas as to what I think the probable cause is and what I conceive I should be doing to change the



#### Building an operating system is child's play . . .



#### .. with Wendin's Operating System Toolbox™

We know.

We used it to build PCVMS, our \$99 version of the versatile VAX/VMS operating system for the IBM PC.

And PCUNIX, the only operating system that puts the powerful features of UNIX on the PC for under \$100.

But we didn't create this powerful software construction set just for us. The Toolbox is for any programmer who wants to build his or her own multitasking, multiuser operating systems.

Systems compatible with the PC-DOS file system and with most MS-DOS programs.

In nine basic modules, Toolbox provides everything you need (including source and object code) to build a personal operating system that fits you and your work habits — instead of the other way around.

And to help you get started, we've written a step-by-step instruction manual that shows you how to write a shell and link it with the Toolbox kernel.

The only thing we don't provide is a creative imagination. If you've got that, the rest is child's play.

Operating System Toolbox From Wendin. Only \$99.



© Copyright 1986 Wendin, Inc.

The people who make quality software tools affordable.

(MON.-FRI., 8-5 PACIFIC TIME)



**ORDER HOTLINE** 

(509) 624-8088



Ask about our other products for the IBM PC and true compatibles.

#### **PCNXTM**

True multitasking, multiuser operating system similar to AT&T's popular UNIX™ operating system.

#### **PCVMSTM**

Multitasking, multiuser version of DEC's powerful VAX/VMS operating system. Runs most MS-DOS programs.

#### XTC®

The ultimate programmer's editor. Multitasking macro language plus multiple linkable windows and buffers.

#### All products priced at \$99 with source code included.

#### DEALER INQUIRIES WELCOME

Foreign orders inquire about shipping.
Domestic orders add \$5.00/1st item, \$1.00 each additional item for shipping, handling, and insurance. We accept Visa/MC, American Express, COD, and Bank Drafts drawn on U.S.

Washington residents add 7.8% sales tax. MS is a trademark of Microsoft, PC-DOS is a trademark of IBM. UNIX is a trademark of AT&T. VAX/VMS is a registered trademark of Digital Equipment Corporation.

Wendin and XTC are registered trademarks of Wendin, Inc. PCNX, PCVMS, Operating System Toolbox, and Personal Operating System are trademarks of Wendin, Inc.

Circle no. 112 on reader service card

situation. I will be watching with interest to see what other readers have to say about this problem. Name withheld by request

#### Avoid Woe when Upgrading MS-DOS

Dear DDJ,

Allen Holub's "A Tale of Woe" (C Chest, September 1986) sparked some vivid memories of upgrading from MS-DOS 2.11 to MS-DOS 3.10. MS-DOS 2.11 and earlier would search for an available file from the beginning of the FAT each time. MS-DOS 3.10 is more efficient because it allocates from wherever it left off. This caused me much grief in trying to change the attributes and delete and replace the two system files Allen mentioned, IBMBIO.COM and IBMDOS.COM for the PC and IO.SYS and MSDOS.SYS for generic MS-DOS. These two files must reside in the first clusters of the disk. The first data cluster is cluster 2. Additionally, they must be contiguous. You can delete and replace these two files by following a few simple rules. The total number of clusters must be no more than the original unless there are some unallocated clusters just beyond these two files. Under MS-DOS 3.10, you can unhide and delete these files and then reboot from a floppy. This will restore the FAT pointer to start searching at the beginning of the disk to modify. You may then copy the system files, which will now be the first n clusters, and they will be contiguous. Now the disk will be bootable. It helps to be able to track through the FAT and directories if there is a problem or if more space is needed. So there may be no need to reformat.

Max G. Heffler Landmark Graphics Corp. 1011 Hwy. 6 S, #120 Houston, TX 77077

#### **OS-9** Continued

Dear DDJ,

I was dismayed to see the thrashing Heitzso gave to the OS-9 operating system in his October letter. I disagree with both his specific examples and his general conclusions (please pardon the assumption of gender).

To start with, OS-9 cannot compete

with Unix on disk speed; Unix keeps part of its file structure in memory whereas OS-9 always keeps its sector allocation bit map on the disk up to date. There is a clear speed advantage to accessing information in memory rather than on disk, but you pay a price in vulnerability. In this case, Microware chose to make the file structure robust and corruptionproof.

Heitzso illustrates the "real problems" he has had with OS-9 by describing his difficulty in using the *tsleep()* function. After reading his letter I wrote a simple C program to test the *tsleep()* function, and I was unable to make it malfunction for any number of ticks I specified, over a range of 1 to several thousand. In every case the timing was +/- one tick, just as specified.

The tsleep() function accepts a single parameter indicating the number of ticks to sleep. Most OS-9 systems use a tick granularity of 1/100 second, which also happened to be the length of time Heitzso wished the task to sleep. I suspect that he requested that the task sleep for one tick; however, the documentation clearly states that a tick parameter of 1 causes the calling task to give up its present time slice. At the expiration of the time slice, the task will be put back on the active process queue, where it will compete for CPU time with other processes that are ready to run. If the calling task gives up its time slice near the end of a quantum and there are no other executable processes, it is quite possible that tsleep() will return after an interval as short as 1/3,000 second.

Heitzso also describes how the OS-9 Format utility has a bug that prevents the user from specifying a cluster size greater than one sector. In reality, the documentation for the Format utility mentions that at present only a cluster size of one is supported!

I have yet to uncover a bug in the operating system. Microware's latest product discrepancy report lists a single bug in the operating system components, and it is triggered by an obscure condition in a little-used system call.

I disagree with Heitzso's conclusion that Microware's customer support is poor. I have found Microware to be quite reasonable in the dealings

I have had with it. Microware provides bug lists and work-arounds to those who request them, and it offers a special telephone hot line for professional software developers. One of my gripes is that the hot line is too expensive, but I have heard that this may change. I hope so.

During a time when many experts in the computing field were touting the use of high-level languages as the best way to create an operating system, Microware quietly crafted an elegant, modular, and extensible gem in assembly language. I look forward to seeing OS-9 dominate the 68000 market as more people recognize its merits.

Kurt Liebezeit Ordinate Systems 505 W. Springfield Champaign, IL 61802

We didn't do our homework when we published Heitzso's letter. This issue contains a look at the OS-9 operating system by Brian Capouch—eds.

#### Correction

Dear DDJ,

The Microsoft-supplied correction to Version 4 of the Microsoft Macro Assembler that Ray Duncan gives on page 96 of the September 1986 issue of *DDJ* is incorrect. The "correction" as published causes MASM to go off into never-never land. The error in the listing on page 96 is a typographical one. The byte entered into address xxxx:72D4 should be E9 instead of 39. This error appears in the string of 34 bytes that are entered starting at xxxx:72B8.

Robert C. F. Bartels P.O. Box 2240 Ann Arbor, MI 48106

DDJ



#### **ANOTHER PLUS FROM** BLAISE COMPUTING

The best just got better! Turbo POWER TOOLS, acclaimed as the best programmer support package for Turbo Pascal, now has

even more functions, more detailed documentation and more sample programs.

#### **NO SECRETS**

Turbo POWER TOOLS PLUS is crafted so that the source is efficient, readable and easy to modify. We don't keep secrets! We tell you exactly how windows are managed, how interrupt service routines can be written in Turbo Pascal, and how to write memory resident programs that can even access the disk. Maybe you've heard of some undocumented DOS features that resident programs use to weave their magic. Turbo POWER TOOLS PLUS documents these features and lets you make your own magic!

Here's just part of the PLUS in Turbo POWER TOOLS PLUS:

- WINDOWS that are stackable, removable, with optional borders and a cursor memory.
- FAST DIRECT VIDEO ACCESS for efficiency.
- SCREEN HANDLING including multiple monitor and EGA 43-line
- POP-UP MENUS which are flexible, efficient and easy to use, giving your applications that polished look.
- INTERRUPT SERVICE ROUTINES that can be written in Turbo Pascal without the need for assembly language or inline code.

#### Memory Resident Routines. Routinely.

◆ INTERVENTION CODE lets you develop memory resident applications that can take full advantage of DOS capabilities. With simple procedure calls, you can "schedule" a Turbo Pascal procedure to execute either when a "hot key" is pressed, or at a specified time.

Window Routines.

- PROGRAM CONTROL ROUTINES allow you to run other programs from Turbo Pascal, and even execute DOS commands.
- **◆ MEMORY MANAGEMENT** allows you to monitor, allocate and free DOScontrolled memory.
- DIRECTORY AND FILE HAN-**DLING** support to let you take advantage of the newer features of DOS including networking.
- STRING procedures allowing powerful translation and conversion capabilities.
- ◆ FULL SOURCE CODE for all included routines, sample programs and utilities.
- DOCUMENTATION, TECHNICAL SUPPORT and attention to detail that

have distinguished Blaise Computing over the years.

Turbo POWER TOOLS PLUS supports Turbo Pascal Version 2.0 and later and is just \$99.95.

Another quality product from Blaise Computing: Turbo ASYNCH

A new package which provides the crucial core of hardware interrupt support needed to build applications that communicate. ASYNCH PLUS offers simultaneous buffered input and output to both COM ports at speeds up to 9600 baud. The XON/XOFF protocol is supported. Now it also includes the "XMODEM" file-transfer protocol and support for Hayes compatible modems.

The underlying functions of Turbo ASYNCH PLUS are carefully crafted in assembler for efficiency and drive the UART and programmable interrupt controller chips directly. These functions, installed as a runtime resident system, require just 3.2K bytes. The high level function are all written in Turbo Pascal in the same style and format as Turbo POWER TOOLS PLUS. All source code is included for just \$99.95.

#### BLAISE COMPUTING INC.

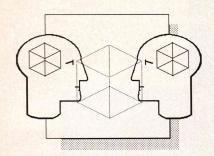
2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

ORDER TOLL-FREE 800-227-8087

	Calif. residents call (415) 540-5441
	the PLUS I need! Enclosed is \$for
1	domestic orders add 510.00 Phone: ( )
1	Name: Zip: Zip: Zip: Zip: Zip: Exp. Date: Zip: Exp Zip: Zip: Exp Zip: Zip: Zip: Zip: Zip:
	VISA or MC #:

Reprinted from PC Magazine, June 10, 1986 Copyright © 1986 Ziff-Davis Publishing Company

#### VIEWPOINT



#### **Logic and PROLOG**

The touted virtue of PROLOG is that it provides a basis for programming in logic—hence its name. This suggests that logically correct descriptions or axiomatizations of a body of knowledge can be transcribed into PROLOG and that the appropriate deductions could then be drawn by PROLOG's inference engine. The idea of systematization of knowledge by way of postulates embedded in a deductive system has proven to be a powerful one since the time of Euclid. With the development of predicate logic by Frege and Russell, the idea received new impetus in this century. Many areas of mathematical and scientific investigation have axiomatic foundations-set theory being a prime example. The expansion and codification of knowledge by the deductive-axiomatic method, as the central methodology of knowledge, has its critics. But what has delayed its use outside theory construction itself is that, for practical real-time applications, hand-deduction from a knowledge base is too

#### by Dick Butrick

slow. What PROLOG promises is speed of deduction. You could simply query a knowledge base (axioms, postulates), and PROLOG would make deductions with computational speed—giving tremendous impetus to the deductive-axiomatic method as the central methodology of knowledge by removing the practical barrier to its use. Unfortunately, the match between

Dick Butrick, Ohio University, Athens, OH 45701. Dick is a professor in the Computer Science Department.

formal logic and PROLOG is tenuous at best. In fact, from a strictly logical point of view, PROLOG is inconsistent. Because any inconsistent deductive system is complete (if you can derive a contradiction, you can derive anything), PROLOG is theoretically complete. In implementation, however, PROLOG is not just inconsistent, it is incomplete. Time considerations and stack space are not considerations in pure logic, but these realities render PROLOG radically incomplete.

Of course, there are PROLOGs and PROLOGs. Specific reference here is to PLS PROLOG. However, the points raised apply just as well to Borland PROLOG and indeed to any nonpure, expanded, DEC-20-type PROLOG (Quintus PROLOG, CPROLOG, Poplog, and so on).

Typically, in testing out PROLOG as a deductive-axiomatic shell, a logician might enter postulates for the transitivity of *R*:

(x)(y)(z)(Rxy & Ryz --> Rxz)

In the Simple syntax of PLS, this becomes:

R(x z) if R(x y) and R(y z)

Given the R-facts  $R(a \ b)$  and  $R(b \ c)$ , it seems reasonable to query the system with  $is(R(a \ c))$ ? PROLOG promptly responds with "no more space." Stack overflow has occurred.

This is not apt to impress the logician, or for that matter the layman, with the deductive power of PROLOG. At this point the dismayed logician might enter the PROLOG equivalent of  $\{p < --> q, p\}$  and query the system for q. The PROLOG equivalent is:

p if q q if p p

And the query is is(q). "No more space" is again the reply.

At this point the logician might wonder if the inference engine is out of gas. In fact the system is in a goal-reduction loop. It reasons thus: To get q, first get p; to get p, first get q; to get q, first get p; and so forth. It never gets beyond the first two rules. In fact, p

*if p, p*} along with the query for *p* will put the system into an infinite loop.

This might seem a simple problem to solve, which of course it is for specific cases. In general, however, it can be shown to be unsolvable. A loop-detection procedure for a goal-reduction theorem prover is equivalent to the halting problem shown by Alan Turing in the 30s to be unsolvable. Essentially, a loop-detection monitor requires more logic than the goal reduction it monitors.

To make matters worse, the "no space left" response can be triggered without setting up a goal-reduction loop:  $\{H(x \ y) \ if \ H(x \ x), \ H(a \ a), \ H(b \ b)\}$  along with the query  $is(H(a \ b))$  causes stack overflow. In Borland PROLOG,  $\{H(x \ y \ if \ H(y \ x), \ H(a \ b)\}$  along with the query for  $H(b \ a)$  does the trick. If you cannot enter the postulates declaring the commutativity of a relation, then you might question whether PROLOG belongs in the remedial logic class for absolute dummies.

Examples such as the foregoing are legion, but they merely demonstrate the radical incompleteness of PROLOG. What is even more insidious is PROLOG's inconsistency. The treacherous dimension to PROLOG is not so much what it can't do as what it can. Consider the following deduction, based on the deduction rule known to logicians as mirabile dictu:

p if not q q / Hence not p

Querying this little postulate set with  $is(not \ p)$  yields the astounding answer "yes." Not only that, the system will make further deductions on the basis of this fallacious deduction. Adding the postulate r if not p and querying the system for r yields the answer "yes." Again, such examples are legion.

Borland refers to PROLOG's treatment of negation as "novel." The justification for this novel treatment of negation is, according to Borland, the hidden assumption of the law of the excluded middle automatically made by PROLOG. Thus for the one-place predicate h, PROLOG automatically assumes h(x) or not h(x). The latter is

(continued on page 138)

#### COMMON LISP Development System for Your PC or AT

#### Introducing TransLISP PLUS", The Consultant's LISP Over 400 Primitives, a C Interface, and Optional Runtime System

People call you because you're an expert. Your customers want to keep up with what's going on. They want software that is more intelligent and responsive, or software that does something that just wasn't possible before. So they call you.

Now you can write and deliver a whole new category of software with TransLISP PLUS, a practical, efficient LISP system. You can also add Artificial Intelligence to your software. We include several features, like a C Interface and Optional Runtime System, so you can control the performance and security of your programs. And your users only need to have a PC (can even be non-compatible) with 320K and 1 floppy drive. Now, what could you write for a \$700 PC? . . . or a \$7000 PC? . . .

#### Add AI Technologies to Your Software

Most of the programs you write are accessed by a user who doesn't have your expertise. Use intelligent interfaces to make your programs more responsive to the end user.

You can even use the C Interface included with TransLISP PLUS to customize LISP, or combine C functions with LISP programs.

Take advantage of AI technologies to make your programs smarter and more flexible.

#### **Extensive Development Environment** Over 400 Primitives

TransLISP PLUS provides you with over 400 primitives for development, including extras for hardware support and operating system access. Their spectrum ranges from control constructs, macros, and special forms, to multi-dimensional arrays, reader support for binary, octal, and hex constants, improved list processing, and system interrupts.

DOS commands and applications can be invoked from within TransLISP PLUS, as can the fast editor. Of course, you can use your own editor if you like.

A variety of debugging tools are provided. The trace facility tracks the evaluation of any built-in or user-defined function or macro.

Traceback, Break, Cross Reference, and Pretty Printer are also provided to help you spot problems.

Over 400 COMMON LISP Primitives

Optional Runtime (No Royalties)

Supports IBM PC color graphics

Supports 8087 math coprocessor

Over 30 Demo Programs with Source

Many Debugging Utilities

Microsoft Mouse Support

Interpreter

**Program Editor** 

#### The ONLY Full Featured Common Lisp with a C Language Interface

The best of both worlds. The interface to Microsoft C gives you a powerful extension to TransLISP PLUS - now you can write code in LISP and C. And you don't need an AT, it will run on your PC!

The C Interface makes it practical for you to write a C program and add it as a new function to TransLISP PLUS. Your function can:

- · extend and/or change the LISP syntax
- · be an entire system of programs

Create your own BUILT-IN primitives which are directly tied to the system and called at full speed by the interpreter. Extend the functionality of your program by including features of your own like macros, functions, and special forms.

Code from C libraries produced by other vendors can be integrated into your program to perform tasks not normally part of LISP.

#### Use PLUS for Your Applications. No Royalties.

Once you own TransLISP PLUS, you may want to use it to distribute applications. No

The Optional TransLISP PLUS Runtime supplies you with a special interpreter. You can distribute an executable version of your program without distributing source code.

The Runtime is available for \$150 and Trans-LISP PLUS is required.

Complete Manual with Tutorial, Indexed

Reference Manual, and Quick

• C Language Interface

Reference Card

· Lexically scoped

· System Interrupts

Use your C Libraries

NOT COPY PROTECTED

· Online Help

#### Don't Know LISP?

Get a solid understanding of LISP with the comprehensive, easy-to-understand tutorial. Each section walks you through a new concept and reinforces it with examples - both text and online

Over 30 demo programs supplement the tutorial. Use them for an in-depth introduction to LISP programming techniques. Commented source is included so you can see how and why the program operates.

The demos cover a wide variety of applications including: Select a word processor, Read dBASE SDF files, Job Counselor, and many, many more.

#### The Fundamentals

If you are interested in learning LISP but don't need all the extras in TransLISP PLUS, order TransLISP for \$95. It's a full, easy-to-use introduction to LISP that includes the tutorial and demo programs described above, and over 300 primitives.

It is a solid subset of COMMON LISP: and you can write programs of up to 12000 lines.

#### **COMMON LISP Standard**

Programs written carefully with TransLISP PLUS will be completely "portable" to any other COMMON LISP system on a micro, mini, or mainframe computer. This allows you, for example, to write a program with TransLISP PLUS on your PC at home, and compile and run it on the VAX at work

#### **System Requirements**

TransLISP PLUS requires at least 320K RAM and a 360K disk

TransLISP requires 256K RAM and a 360K disk drive.

TransLISP PLUS		\$195
TransLISP		\$ 95
Upgrade TransLISP to PLUS		\$158
TransLISP PLUS Runtime		

#### TO ORDER OR FOR DETAILS CALL



300-821-2492 VISA

Use TransLISP PLUS to program with and deliver to lots of machines . . . Use your existing Clibraries . . . Distribute your applications

#### MONEY BACK GUARANTEE

Try TransLISP PLUS (\$195) for 30 days — if not satisfied get a full refund.



335-D Washington Street, Norwell, MA 02061, (617) 659-1571

TransLISP and TransLISP PLUS are trademarks of Solution Systems. Solution Systems is a trademark of Solution Systems

Circle no. 148 on reader service card.

# 680xx Computers: Where Are They Going?



#### by Nick Turner

ince the beginning of the microprocessor industry, two groups of programmers have emerged—a result of the divergence between those who liked the 8080 and those who liked the 6502. The 8080 had dedicated I/O instruc-

The 68000 led to the first powerful sixer machine.

tions that used a separate address space, and it had several fairly specialized internal registers. Its instruction set was designed to be powerful and specific. To learn it programmers had to memorize its specialized instructions and address modes. Conversely, the 6502's instructions were more general purpose but less powerful. The instructions were easier to learn, but it took more of them to accomplish the same tasks. Its I/O was memory-mapped, making input and output identical (from the processor's point of view) to normal memory addressing. Memory-mapped I/O also reduced the number of specialized instructions that had to be learned. The two camps began to diverge as early as 1974, and by the early 80s, those programmers who were most fervent acquired the nick-names "eighters" and "sixers."

When the Z80 was introduced, the eighters suddenly had a much more powerful tool. The sixers had the 6800, and then the 6809, but there weren't many viable machines that used those chips. Ardent sixers had to be content with their Apple IIs and Commodore Pets and 64s. Certainly, some really magnificent work came out on the Apple II during the early 80s, showing that the Volkswagen of personal computers was a lot more powerful than had been thought. Toward the end of this period, some manufacturers tried to get the best of both worlds—

Nick Turner, 501 Galveston Dr., Redwood City, CA 94063. Nick is a DDJ editor. for example, the OSI Challenger III, a strange machine with a Z80, a 6502, and a 6800 all in the same box.

For several years, it seemed likely that the Intel line of CPU chips would eventually take over the market. Many devoted

sixers shuddered when the IBM PC turned out to have an eighter chip as its heart.

It wasn't until the 68000 appeared that there was a really powerful sixer machine. What led to the development of the 68000? Motorola's engineers perceived the need for a much more general instruction set so that the chip design could be cleaner and easier to mask. They wanted a CPU that would be upward compatible with future, more powerful chips, without the need for expensive "modes" that hamper functionality and take up space on the chip. For the most part, they seem to have succeeded. Though the 68000 does have some disadvantages and departures from a truly general-purpose design (such as the inability to store data using PC-relative addressing), it's a far cry from the restrictive modes and specializations of the high-end eighter chips. Motorola took an enormous gamble in introducing what was projected to be a whole series of CPUs with a completely new instruction set. Lowlevel programs had to be rewritten for the 68000. There was (and still is) a lot of momentum in the Intel line of chips. Has Motorola's gamble paid off?

#### Today's 680xx Line

To me the most valuable feature of the 68000 line, aside from the philosophy behind it, is the enormous range of speed and power available. With few or no changes to your software, you can move up from the 68008, which is roughly comparable to a fast Z80 in power, all the way to



are currently made by Motorola, but a growing number are being designed by other companies expressly for the 68000 line (an indication of the health of the 68000 standard). The most important support chips are the 68851 MMU and the 68881 FPU, both of which are true coprocessors when used with the 68020 (or later) CPU.

Perhaps as a result of the ease of designing with the 68000, a new flock of 68000-based computers has appeared in the last three years. Table 1, page 18, shows a sampling of the range of power and speed in the line. Of course, there are the relatively low-price personal systems, most of which have graphics-oriented interfaces. But you also have high-end workstations, such as the Sun systems and the Apollo Domain network, and a host of other high-end systems, including some powerful image processing equipment. The VME bus, originally designed around the 68000 line, has rapidly become a worldwide standard for rugged, powerful, modular computer systems. It's supported by an international coalition of companies, the VME International Trade Association (VITA), and is one of the most carefully defined system specifications I've encountered.

Some of the most interesting systems are the most recent. For example, the Mustang-020, a 68020-based system, has been available since last spring. It's amazingly powerful for its price, and it shows great promise as a general-purpose industrial machine. One of its chief assets is its size—in a box no larger than an IBM PC, it provides more power than do many minicomputers. Another system to watch is the Quantum QL, which comes from Sinclair in England. According to some preliminary literature, the entire computer is contained within a box the size of an IBM PC keyboard, and it runs a multitasking operating system called QDOS.

well with multitasking operating sys-

tems-for example, Unix runs well under the 68020. Another interesting operating system that has recently been increasing in popularity is OS-9. Originally designed for the 6809, OS-9 is fast, small, and most important, highly accessible. (See Brian Capouch's article on OS-9 on page 30.)

The 68000 line also lends itself well to graphics-oriented interfaces because of its efficient handling of large memory spaces and its ability to easily accommodate large memory-mapped I/O spaces. The Macintosh operating system, for example, has set a new user interface standard for personal computers. Although the Mac OS (especially in its original incarnation) contained some major flaws, particularly in the area of disk organization, it has been much imitated, and most of the flaws have been corrected with the release of Apple's Hierarchical File System. The Amiga and Atari ST both have similar "desktop" screens, icons, and mice.

In the area of languages, the most prominent has been C. Table 2, page 18, illustrates some typical execution times of programs compiled in C on 68000-based machines. The benchmarks in the table were chosen to illustrate a variety of operations roughly representative of actual programs. C is in some ways perfectly matched to the 68000 line-the regularity and generality of the instruction set makes a language such as C almost a natural. Because of its relatively low-level nature, C translates readily into well-defined groups of machine instructions.

#### The Future

The 68000 line is the most serious competition for the Intel line, and the machines that use 68000s are varied and colorful. But will the 68000 line continue to grow and diversify? Will the industry support new members by

creating products that incorporate them? I think so, for several reasons.

First, Motorola seems to have latched firmly to the idea that object-code compatibility is a must in the 68000 line. So far, each of the new CPUs has been almost completely compatible with its predecessors. The only exceptions come when you're writing time-slicing OS code or you have to deal with interrupts. Typically, that sort of code is fairly easy to upgrade, and the actual applications (if they're written intelligently) seldom require any changes.

Second, you can expect continued growth and diversity in the 68000 line. Some preliminary information on the 68030 processor has just landed on my desk. It will have not only the instruction cache of the 68020 but also a data cache, a memory manager, and much more. The result will be a CPU that has the 68020's speed in tight loops (because of the instruction cache) and that also can at times execute completely on-chip for extended periods (because frequently accessed data will be in the cache). The rumored 68040 may use a 64- or 128-bit data bus for reads (although it will still use a 32-bit bus for writes). It may contain an even larger RAM cache as well, plus a

more powerful memory manager. In the peripherals department, the 68882 will be a more powerful, pin-compatible version of the 68881 floating-point coprocessor that, because of internal multiprocessing, will significantly outstrip the performance of its predecessor.

Finally, there are rumors of many other support chips floating around, along with some really fun rumors about 68100 or 68200 CPU chips. For example, how about a chip containing four 68030 equivalents, all executing simultaneously from dual, dynamically allocated 16K instruction and data caches? Of course, that's nothing more than a nasty rumor. There's probably no truth to it whatsoever.

#### A Dream Machine?

Will future developments live up to the expectations of software developers? So far the 68000 line has done well in a market that might otherwise have been completely dominated by Intel. The 80386 is strong competition, but the 68030 sounds like a programmer's dream. Unless some company takes over the sixer marketplace within the next few years, it looks like the 68000 and the more powerful related chips will prevail as the premier sixer CPUs.

**DDJ**Vote for your favorite feature/article.

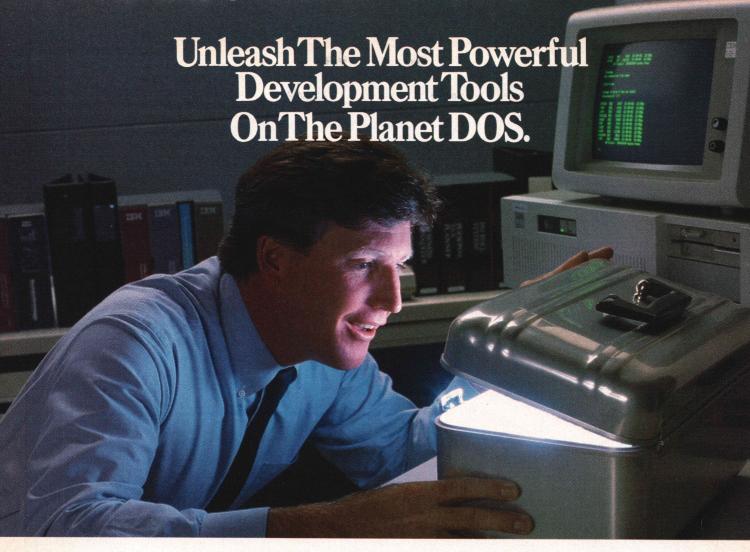
Circle Reader Service **No. 2.** 

Туре	Maker	Model	Base Prices	Memory Range (Mbytes)	CPU	Clock Speed (MHz)	os	Open Design	Intro Date
Portable	Sinclair	QL	\$266	.5	68008	8?	QDOS	No	1986
Macalikes	Atari	ST 520	\$800	.5-1	68000	8	TOS	No	7/85
	Commodore	AMIGA 1000	\$1,000-\$3,000	.25-4	68000	7.16	AmigaDOS	Yes	10/85
	Apple	Macintosh Enhanced	\$1,700	.25-4	68000	7.8	Mac OS	No	4/86
		Macintosh Plus	\$2,200	1-4	68000	7.8	Mac OS	No	1/86
Tile Consum		"Jonathon"	?	2-?	68020?	16?	Mac OS? Unix?	Yes	3/87?
Workhorses	Data-Comp	Mustang-020	\$4,000	2	68020	12-16	OS-9	No	3/86
4. 图 图点	Various	VME bus	\$4,000-\$20,000	.5-16	(various)	8-20	(various)	Yes	
Workstations	Apollo	Domain	\$9,900-\$70,000	2-16	68020	12-20	Unix	Yes	2/86
	Sun	Sun II	\$19,900	1-4	68010	10	Unix 4.2	Yes	11/83
		Sun III	\$19,900	4-16	68020	16.7	Unix 4.2	Yes	9/85

Table 1: Comparison of representative 680xx systems

Benchmarks:						
System	Compiler	Looptst (500)	Pointer (1500)	Fibtest (18)	Sieve (140)	
Amiga	Lattice	44.3	37.5	48.8	81.7	
	Manx Aztec 16	29.4	33.3	35.1	64.3	
	Manx Axtec 32	38.4	35.1	40.1	80.7	
Atari ST	Aleyon	25.6	28.4	31.2	56.5	
	Lattice	30.7	30.7	35.0	62.3	
	Mark Williams	30.7	30.0	37.2	60.1	
	Megamax	25.6	35.3	33.3	53.9	
Macintosh	Lightspeed	37.4	42.3	45.4	78.9	
IBM PC	Microsoft C	25.8	30.9	37.1	60.1	

Table 2: Representative C compiler benchmarks (in seconds)



#### UNIFY DBMS/DOS. The UNIX World Leader Brings A New Dimension To DOS Application Development.

What happens as the DOS world expands? As a new generation of hardware takes over? As networking becomes more important? The potential is enormous. But until now, the tools to achieve it have been limited.

Now a leader from another world unleashes that potential: UNIFY® DBMS. The leading relational DBMS in the UNIX™ world. And now, the most advanced set of application development tools in the DOS world.

With UNIFY DBMS, DOS developers have new power to build more sophisticated applications than ever before possible.

The power to write high performance "C" programs that will access the data base, using Unify's Direct Host Language Interface.

The power of an industry standard

query language-SQL.

The power of unmatched speed in production applications. Only UNIFY DBMS is specifically engineered for transaction throughput. With unique performance features like PathFinder™ Architecture multiple access methods, for the fastest possible data base access.

The power of comprehensive program development and screen management tools. Plus a state-of-the-art fourth generation

report-writer.

What's more, with UNIFY DBMS, the potential of networked applications becomes a reality. Unlike DBMS systems which were originally single-user (and which have a long stretch to accommodate more users), UNIFY DBMS is a *proven* multi-user system.

And because UNIFY DBMS/DOS is the best of two worlds, it offers you the most powerful benefit of all: DBMS applications that can grow as your needs grow. From single user DOS. To networked DOS. To multi-user UNIX. All without changing your applications.

Call the Unify Information Hotline for our free booklet: The New DOS World. (503) 635-7777



4000 Kruse Way Place Lake Oswego, OR 97034

#### "How to protect your software by letting people copy it."

By Dick Erett, President of Software Security



Inventor and entrepreneur, Dick Erett, explains his company's view on the

protection of intellectual property.

crucial point that even sophisticated software development companies and the trade press seem to be missing or ignoring is this:

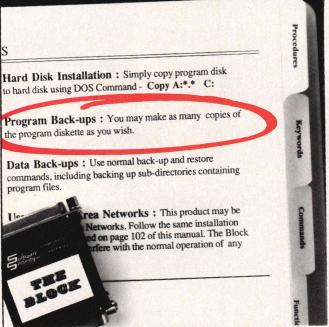
Software protection must be understood to be a distinctively different concept from that commonly referred to as copy protection.

Fundamentally, software protection involves devising a method that prevents unauthorized use of a program, without restricting a legitimate user from making any number of additional copies or preventing program operation via hard disk or LANs.

Logic dictates that magnetic media can no more protect itself from misuse than a padlock can lock itself.

Software protection must reside outside the actual storage media. The technique can then be made as tamper proof as deemed necessary. If one is clever enough, patent law can be brought to bear on the method.

Software protection is at a crossroads and the choices are clear. You can give product away to a segment



Soon all software installation procedures will be as straightforward as this. The only difference will be whether you include the option to steal your product or not.

of the market, or take a stand against the theft of your intellectual property.

#### "... giving your software away is fine..."

We strongly believe that giving your software away is fine, if you make the decision to do so. However, if the public's sense of ethics is determining company policy, then you are no longer in control.

We have patented a device that protects your software while allowing unlimited archival copies and uninhibited use of hard disks and LANs. The name of this product is The BLOCK™.

The BLOCK is the only patented method we know of to protect your investment. It answers all the complaints of reasonable people concerning software protection.

In reality, the only people who could object are those who would like the option of stealing your company's product.

#### "...eliminating the rationale for copy-busting..."

Since The BLOCK allows a user to make unlimited archival copies the rationale for copy-busting programs is eliminated.

The BLOCK is fully protected by federal patent law rather than the less effective copyright statutes. The law clearly prohibits the production of work-alike devices to replace The BLOCK.

The BLOCK attaches to any communications port of virtually any microcomputer. It comes with a unique customer product number programmed into the circuit.

The BLOCK is transparent to any device attached to the port. Once it is in place users are essentially unaware of its presence. The BLOCK may be daisy-chained to provide security for more than one software package.

Each software developer devises their own procedure for accessing The BLOCK to confirm a legitimate user. If it is not present, then the program can take appropriate action.

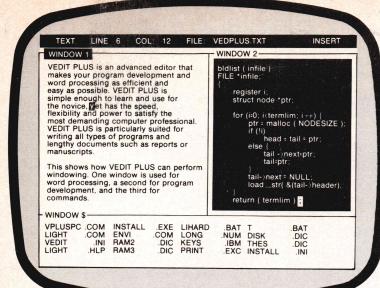
#### "... possibilities... limited only by your imagination..."

The elegance of The BLOCK lies in its simplicity. Once you understand the principle of The BLOCK, hundreds of possibilities will manifest themselves, limited only by your imagination.

Your efforts, investments and intellectual property belong to you, and you have an obligation to protect them. Let us help you safeguard what's rightfully yours. Call today for our brochure, or a demo unit."

Software ecurity inc.

870 High Ridge Road Stamford, Connecticut 06905 203 329 8870





# CHOICE IN PROGRAMMABLE EDITORS

VEDIT PLUS has been the #1 choice of professionals since 1980. Our latest release is even better - you can open windows to simultaneously edit several files, access many editing functions with pop-up menus, use keystroke macros to speed editing, and run other programs or DOS commands from within VEDIT PLUS.

Whether your needs are program development, technical writing or word processing, VEDIT PLUS is your answer. With over 40,000 users you can depend on VEDIT PLUS to perform consistently and reliably. It is simple enough to learn for the novice, yet has the speed, flexibility and power to satisfy the most demanding professional.

Compare. No other editor is as powerful - unlimited keystroke macros, instant 'off-the-cuff' command macros utilizing a complete programming language, single command file comparison, special word processing and programming features. No other editor is as easy to use - on-line help, pop-up menus, 75 page tutorial, 380 page manual, and VEDIT PLUS is completely customizable.

Fully supports color windows on IBM CGA & EGA, and even windows on most CRT terminals (including CRT's connected to an IBM PC). Available for IBM PC, TI Professional, Tandy 2000, DEC Rainbow, Wang PC, MS-DOS, CP/M-86 and CP/M-80. Order direct or from your dealer. \$185

"To sum things up, VEDIT PLUS is a small, fast, sophisticated editor with a wealth of features and a good macro language. It offers many rewards for the dedicated programmer."

#### Computer Language, Chris Wolf, Scott Lewis, Mark Gayman 6/86

"VEDIT PLUS is a wholly remarkable program: blindingly fast, extremely powerful, and highly flexible."

Profiles Magazine, Robert Lavenda 4/86

#### **VEDIT PLUS FEATURES**

- Simultaneously edit up to 37 files of unlimited size.
- Split the screen into variable sized windows.
- 'Virtual' disk buffering simplifies editing of large files.
- · Memory management supports up to 640K.
- Execute DOS commands or other programs.
- MS-DOS pathname and CP/M user number support.
- Horizontal scrolling edit long lines.
- · Flexible 'cut and paste' with 36 text registers.
- Customization determine your own keyboard layout, create your own editing functions, support any screen size, any CRT.
- Optimized for IBM PC/XT/AT. Also 132 column & up to 70 lines.

#### **EASY TO USE**

- Interactive on-line help is user changeable and expandable.
- On-line integer calculator (also algebraic expressions).
- Single key search and global or selective replace.
- Pop-up menus for easy access to many editing functions.
- Keystroke macros speed editing, 'hot keys' for menu functions.

#### FOR PROGRAMMERS

- Automatic Indent/Undent for 'C', PL/I or PASCAL.
- Match/check nested parentheses, i.e. '{' and '}' for 'C'.
- Automatic conversion to upper case for assembly language labels, opcodes, operands with comments unchanged.
- Optional 8080 to 8086 source code translator.

#### FOR WRITERS

- Word Wrap and paragraph formatting at adjustable margins.
- Right margin justification.
- Support foreign, graphic and special characters.
- Convert WordStar and mainframe files.
- Print any portion of file; separate printer margins.

#### MACRO PROGRAMMING LANGUAGE

- 'If-then-else', looping, testing, branching, user prompts keyboard input, 17 bit algebraic expressions, variables.
- CRT emulation within windows, Forms entry
- Simplifies complex text processing, formatting, conversions and translations.
- Complete TECO capability.
- Free macros: Full screen file compare/merge Sort mailing lists Print Formatter Main menu

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. MS-DOS is a registered trademark of Microsoft. CP/M is a registered trademark of Digital Research. WordStar is a registered trademark of MicroPro.

Circle no. 122 on reader service card



# A Mini Forth for the 68000

by G. Yates Fletcher

y exposure to Forth has been limited to the proselytizing of a few Forth fanatics, articles in various publications, and a fairly careful reading of Leo Brodie's excellent book Starting Forth (Englewood Cliffs, N.J.: Prentice-Hall, 1981). One of the attractions of the language for me is its fundamental simplicity. The interpreter, threader, and kernel of basic supporting words are small enough that (at least in theory) a single programmer can create a working system with a moderate amount of effort. I often toyed with the notion of doing it myself just for the learning experience (read that fun), but I never got to the point of making a serious attempt until last fall when I was assigned to teach for the umpteenth time our sophomorelevel course in computer organization and assembly language. I was casting about for some way to generate enthusiasm, which although not a prerequisite for teaching seems to make the process more enjoyable for all concerned. It struck me that building a Forth system might make good

I thought, what the heck, I'll write a "no frills" Forth and give the students the executable code (so they can play with it and see how it's supposed to work) and the source code minus the modules they will be required to write. By the time I've taught them enough about assembly-language programming to do the job

programming exercises for my stu-

dents. Thus was FLINT born.

G. Yates Fletcher, Dept. of Computer Science, North Carolina State University, P.O. Box 8206, Raleigh, NC 27695-8206. Mr. Fletcher is an assistant professor of computer science. The inside approach to Forth makes the language much more palatable.

and enough about Forth that they understand what is required, the semester should be winding down. Amazingly enough, everything went pretty much according to plan (albeit with considerably more sweat than I anticipated). As you might have guessed by now, the instructor probably learned a lot more than the students, but that's one of the reasons why I took the job.

The product of this labor is not a standard Forth. As the venture was educational/recreational and not commercial, I never bothered to find out what the standards were or even look at the code for a real Forth. Thus I have chosen the acronym FLINT, for Forth-like interpreter and threader, to describe the system. FLINT was basically reverse-engineered from Brodie's description of the language, so the implementation is probably a blend of novelty and naiveté. Nevertheless, I feel that it is more than a toy, as I have used it to write a turtlegraphics program for my terminal; a full-screen editor (for Forth screens, of course); and a version of a standard prime-number seive benchmark (Byte, January 1983) that runs in a little more than 20 seconds on an 8-MHz 68000, making it faster than any of the microcomputer Forths listed.

I can conceive several levels at which this program may be useful. For those not familiar with Forth, it might help to be an introduction. It is no substitute for a good book such as Brodie's, but it could make a worthwhile companion. Those approaching Forth from the outside as another language are often put off or puzzled by its many idiosyncracies. The inside approach to Forth as a program gives a complementary perspective that resolves many of these mysteries and makes the language much more palatable. For do-it-yourselfers this program is evidence that novices can indeed produce something workable. They can read enough of the description and documentation to get a good idea of what needs to be done and blast off on their own, referring to the code perhaps when stuck or merely to confirm that their own way of doing things is better. Forth programmers who don't have a version running on their own 68000 machine might be able to revise and polish it up enough so that it resembles some usable standard. Forth cognoscenti might be interested to know that FLINT produces what I have since learned is subroutine threaded code (which seems to be particularly appropriate for the 68000) as opposed to the more usual indirect threaded code.

#### **Overall Structure**

FLINT was written with several goals in mind: to provide students with an example of a well-structured, real world, assembly-language program; to illustrate the utility and power of the 68000 instruction set and addressing modes; and to test the theory that Forth is more naturally understood as a program rather than as a language.

I've made a fairly serious attempt to structure and comment the code properly. Because everything is in there somewhere, much can be learned (at least in theory) by reading the program carefully. In fact, several of my students have intimated (very discreetly) that they found the code much less confusing than my explanations of it. The 68000 assembly code comprises a minimal kernel of less than 500 lines and is arranged as shown in Table 1, below.

FLINT requires some basic BIOS support from the host in the form of macros that users must tailor for their own machines and resident system software. My system is a Sage II (now Stride), which has two floppy drives, 512K RAM, and 64K PROM and runs on an 8-MHz 68000. The PROM contains all the necessary BIOS support as well as a monitor/debugger that furnished an excellent environment for developing and running FLINT. Macro definitions for my system are in Listing One, page 52.

#### Interpreter and Dictionary

The token is the basic unit processed by the interpreter. Tokens are not obtained directly from the terminal but are taken from an 80-character line buffer. After prompting the user, routine LINE (Listing Two, page 52) fills the line buffer from the terminal and terminates upon receiving a carriage return character. TOKEN takes its input from the buffer and recognizes the blank as a token delimiter. The carriage return embedded in the buffer (the one that terminated LINE) is seen as a legal token whose execution returns control to LINE. In support of its buffer-filling function, LINE recognizes and handles backspaces.

The interpreter is composed of several routines whose operation forms an instruction cycle fed by the user interactively. The instructions are small modules of code called words whose actions are normally directed at data (or pointers to the data) residing on a parameter stack. The words are identified by short alphanumeric strings (tokens) that are symbols or English words that usually describe or are related to the action they perform. The words are arranged in a linked list in which each node contains the identifying token, a link pointer, and the code defining the word's action. This linked list is called the dictionary.

A dictionary entry starts with the

identifier, which is a 4-byte value. It consists of the length and the first three characters of the name of the word being identified. For example, if the word were *EXECUTE*, the identifier would contain a 7 and then the characters *EXE*.

The next field is the link pointer, which is a 2-byte field containing the address of the previous word's identifier. Thus a dictionary search always starts with the most recently defined word and works backward. For larger systems you might want to make the link pointer a 4-byte value or perhaps make it a relative value instead of an absolute address.

After the link pointer is the code field, which contains the machine code that runs when the word is executed.

The outer interpreter is a loop that waits for and accepts input tokens, searches the dictionary until a match is found, and extracts the address of the corresponding code. The address can be sent to the inner interpreter for direct execution. If a token is not in the dictionary, it is assumed to represent a number in the current base, and its value is extracted and placed on the stack. If this attempt fails, an error is assumed and the WHAZZAT token is invoked.

Words can be defined as well as executed. The interpreter has an alternate compile mode (the standard mode is execute mode). The interpreter's primary task remains that of extracting the code address of an input token. When in execute mode, a JSR to this address is executed as before. But when the interpreter is in compile mode, a JSR to the code address is written into the dictionary. Execution is deferred until the word being defined is invoked in execute mode. Any number encountered in compile mode is handled by generating code in the dictionary that will push the value onto the stack at execution time. These compiled numbers are called literals. Additional support is provided for a submode in which words can be defined directly in machine code.

#### **How FLINT Works**

For a concrete example of how FLINT works, let's look at the activity that occurs as the first word in the inner shell, *CONSTANT*, is defined. Envision using this word interactively as follows:

10 CONSTANT TEN

or

12 CONSTANT DOZEN

CONSTANT will thus be invoked to define words that are constants. If, for example, you invoke the word TEN, you will expect it to push the number 10 onto the stack; if you invoke DOZ-EN, you expect 12 to be pushed; and so on. Let's call TEN and DOZEN instances of CONSTANT. Thus when CONSTANT is invoked, it will take the value of the instance from the stack and the name of the instance from the input stream.

Now look at the definition of *CON-STANT* to see how this activity is to be directed. The defining string is

: CONSTANT TOKEN HEADER LITERAL CODE 3AFC 4E75;

The outer interpreter, operating in execute mode, picks up the token ":", finds a match in the dictionary, and proceeds to execute it. Examining the code for ":", you see that a subroutine call (JSR) to TOKEN will be excuted. The action of TOKEN, of course, is to pull in the next token from the input stream, and in this case it will be the token CONSTANT. The next JSR is to HEADER, whose action is to produce a dictionary header for the newly cap-

Outer interpreter

Line-buffer input routines: PROMPT LINE

Words supporting execute mode: *TOKEN*, *SEARCH*, *NUMBER*, *EXECUTE*, *WHAZZAT* Words supporting compile mode: *COMPILE*, *HEADER*, *IMMEDIATE*, ":", *CODE*, ";", ",",

LITERAL

System variables: BASE, CBLOCK, EDBUF, DICT

Words supporting disk I/O: LOAD, GO, SAVE, INTERACTIVE

Words supporting terminal output: TYPE, ".", ".S"

Miscellaneous words: "(", QUIT, LOGOFF

Table 1: Arrangement of 68000 assembly code in kernel

68000 MINI FORTH (continued from page 23)

tured token *CONSTANT*. The remaining activity initiated by the definition will produce its code field.

The last activity of ":" is to set the compile flag, FCOLON, and return to the outer interpreter. The interpreter pulls the next token, TOKEN, from the input stream and extracts its code address. Because the interpreter is now in compile mode, a JSR to TOKEN will be written in the dictionarythat is, into the code field of CON-STANT. This means that the first activity of CONSTANT, when it is itself executed (called execution-time behavior), will be (surprise, surprise) to pull in the name of the instance from the input stream. In like manner a JSR to HEADER will become the next part of the code for CONSTANT, so now the execution-time behavior of CONSTANT has been defined up to the point where a header for the instance has been created. It is now time to define the execution-time behavior of CONSTANT that will define the execution-time behavior of its instances. Thus you must direct CON-STANT to take the value of its instance from the stack and write code in the dictionary that upon execution will place this value on the stack. This is an exact description of what LITERAL does, so its inclusion in the definition solves the problem neatly.

The activity CONSTANT must perform that has not yet been coded is to close the definition of its instance. At this point in the definition process, the outer interpreter, in compile mode, picks up the token CODE. A careful examination of the header for CODE shows that its listed token length is 132, which is 128 more than it should be. This is no accident. It is in fact the manner in which words are tagged as immediate, meaning that they are to be executed even when the interpreter is in compile mode. The necessity for such words should be obvious because otherwise, for instance, there would be no good way to terminate a definition. (When TO-KEN picks up an immediate word, it sets the immediate flag, FIMMED, to let the interpreter know that the word is to be executed.) CODE sets the system base to 16 (hex); sets the code flag, FCODE; and returns to the interpreter.

When the tokens 3AFC and 4E75 are picked up, they are not found in the dictionary. Thus each one is sent in turn to NUMBER, which extracts its proper hex value and places it on the stack from where the interpreter (now in code mode) copies them into the dictionary. The code 3AFC 4E75 translates as MOVE.W #04E75H,(A5)+. Thus the final activity in the run-time behavior of CONSTANT is to copy 04E75H into the dictionary definition of the instance. Because 4E75 is the code for RTS, this activity will effectively close the definition of the instance. All that remains is to close the definition of CONSTANT. This is the job of ";", which resets the system base to 10, clears the compile and code flags, and writes an RTS into the dictionary.

Well, there you have it: just a simple little definition. Readers new to Forth (if there are any who have made it this far) probably feel that the claims for its simplicity are highly exaggerated, and even those with some familiarity may find themselves a bit glazed over. The bad news is that the operation of the FLINT compiler is often a fairly complicated activity. The good news is that it seldom gets any more complicated than the example given here. Complexity is after all a relative thing. Would anyone care to write a step-by-step explanation of the operation of a Pascal compiler on a segment of code that invokes most of its major machinery?

For beginners the definition of CONSTANT is as much a puzzle as it is an example. Understanding it requires clear thinking and a careful reading of the code involved. This is in fact one of the major reasons for discussing it. FLINT's interpreter/ compiler is put to work at three levels. At one level the word CONSTANT is being compiled. To understand the activity at this level, however, you must see through to the level at which CONSTANT will be executed. This level itself must be understood in terms of the activity that will occur at the next level-that is, when a particular instance of the word is executed. The activity at all of these levels is mediated by the same interpreter. The code defining its basic structure occupies half a page, and many of the supporting words—for example, EX-ECUTE, COMPILE, HEADER, ":", CODE,

";", and LITERAL—are only two or three instructions long. Finally, if you stand back from the example a little, you see that an entire data type has been created by a nine-word statement. I feel that anyone who understands the simplicity underlying this example has a firm grasp of FLINT and should have no real trouble mastering Forth.

#### The Inner Shell

Once the embryonic FLINT system is running, it is ready for a big meal of nourishing words that will give it more size and power. These inner shell words support:

- definition and manipulation of constants, variables, and arrays
- stack manipulation and stack arithmetic
- structured control for branching and looping

Listing Three, page 58, contains the inner shell words. Many of these words are defined directly in machine code (with the assembler mnemonics given as comments), so you might ask why they are not placed in the kernel and assembled as part of the basic system. My reasons for not doing so are primarily pedagogical. I wanted to maintain as much as possible the purity of the kernel to underscore its elegance and simplicity. In addition, the words, most of which are very short, seem to me to be more readable in Forth format than they are in assembly-language format. This representation, even allowing for comments, is much more compact because the work of building the headers is left to the system.

Structured control in Forth is directed by immediate words, placed in colon definitions to effect the compilation of appropriate conditional branching instructions. The flow of control is conditioned by values on the stack that act as flags. A zero value means false, and any nonzero value means true. As an example let's define the word *ODD*, which takes a number from the stack and prints an asterisk if it is odd:

: ODD 2 MOD IF 42 EMIT THEN;

The action of *IF* when *ODD* is executed is to pop a value (assumed by *IF* to

TRY TO FIND A MORE PACKAGE. VERSATILE WINDOW DEVELOPMENT TOOL

Knoviekte firte

ation this relian

cheke dishibe neie

institution food lyps not had

Lientmer idean ak Cinco Gillita System Others have tried. Finally, a WINDOW tool for the serious C developer. Introducing ASPEN SCIEN-TIFIC'S CURSES WINDOW DEVELOPMENT

- □ UNIX System 5 compatible function library
- Choice of Microsoft, Lattice, Computer Innovations or DeSmet compilers
- ☐ IBM EGA, CGA and monochrome displays as well as special modes for IBM BIOS and Microsoft ANSI driver compatibles (text modes only)
- □ MS-WINDOWS compatible using BIOS mode
- □ Full featured keyboard support
- □ Lightning fast, flicker free memory map output for IBM PC/XT/AT and compatibles
- CURSES library can be ported to the compiler and system of your choice
- Over 115 fully documented functions and
- ☐ Includes auto wordwrap, full color and special extension routines for PCs

What can all these features do for you? CURSES checks display types at run time so your source code does not have to be modified to work with different displays.\* And with BIOS and ANSI support, your screen interface can run on virtually any PC using MS-DOS 2.0 or later. CURSES will enhance your product's human interface and save you time and money!

Let Aspen Scientific's CURSES decorate your

Call and order NOW, only 1000 copies will be sold at the low price of \$89.00.

Requires DOS 2.0 or later, 192K, 1 Disk Drive. With use of run-time video attribute macro directives.



COLORADO 80034-0072 (303) 423-8088

YES! I want my product to come alive with: — copies of CURSES Window Development Package \$ 89 ea. -copies of CURSES including source code Amount Enclosed Amount Enclosed
Price includes shipping to all U.S. cities

Outside USA make payment by International Postal Money Order \$289 ea. My compiler is: Microsoft C 4.0 Microsoft C 3.0 Company Shipping Addres City Telephone State Zip 30 day money back Circle no. 360 on reader service card.

Microsoft, MS-WINDOWS and MS-DOS are trademarks of Microsoft Corporation. IBM is a trademark of International Business Machines Corporation. UNIX is a trademark of Bell Labs. Lattice is a trademark of Lattice, Inc. DeSmet is a trademark of DeSmet Software. Computer Innovations is a trademark of Computer Innovations, Inc.

#### **CLEAN SCREEN MACHINE**

**Data Entry** 

Menus

**Windows** 

**Prototyping** 

**Toolkit** 

# C-scape

#### **■ Total Screen Control**

**C-scape** is a combination screen generator and library of screen I/O functions. Written for C programmers, C-scape brings a fresh approach to the need for an easy-to-learn and use, but truly powerful and flexible screen management tool.

C-scape's kernel is your most powerful ally. Without requiring parameters you'll never use, it allows you to create tailored functions with ease and simplicity. Each key is individually definable. If you know *printf*(), you can use C-scape. C-scape's kernel provides a veritable screen design and construction toolkit to rewrite our functions or to write your own.

#### **■ Most Powerful Prototyping Available**

C-scape offers a unique approach to prototyping your software. You may use **Dan Bricklin's Demo Program** to create, edit, and view your screens (you can even capture existing screens from other programs), and then use C-scape's **demo2c** utility to convert each screen to code.

You can design each screen with attributes such as colors, menu selections, data entry fields (including type, validation, and field naming), masking, and text, and then automatically convert the entire screen to code.

#### ■ Powerful Function Library

Use C-scape's functions for Lotus-like, pull-down, or your own menu designs, automatic scrolling, pop-up windows (number limited only by RAM), logical colors, help, time and date, yes/no, tickertape fields, secure and protected fields, and many others, to turn your demo into a fully functioning and complete program in a fraction of the time spent coding screens from scratch.

C-scape's extensive library includes just about all the data entry and display functions you'll ever need, including money functions, fully definable borders, and orthogonal field movement (get the latest list by calling for more information). And modifying our functions or writing your own is easy. C-scape adjusts automatically for CGA, EGA, monochrome, and the Hercules Graphics Card Plus in RamFont mode, and optionally writes directly to video memory, so it's flexible and fast.

#### **■** Bridges to Power

C-scape includes examples of how to bridge to other powerful tools such as **c-tree** and **db\_VISTA**. You'll be integrating demos to dictionaries to file handlers and database managers in no time. You can even use C-scape to provide the screen design for AI applications, using TransLisp Plus and other packages that support calls to C.

#### **■ Clean, Complete Documentation**

C-scape's documentation is a clear example of how to write for programmers in a hurry. A short introduction uses helpful examples to explain the C-scape design. Each function is documented separately. An index makes reference easy, and a quick-reference card provides a synopsis of each function.

#### ■ Source Code Included/Portable/No Royalties/No Runtime License

Providing source code at no additional cost gives you the freedom to modify existing functions without raising cost as a barrier. The source code includes all the low level routines you might need to port C-scape to an unsupported machine or compiler. Speaking of barriers, you pay no royalties or runtime license fees, either.

#### ■ Toll Free Support and Bulletin Board

To make your job even easier, we provide technical support (toll free), and access to our 24-hour Bulletin Board.

#### **■ Easy Prices/Risk-Free Terms**

Try C-scape for 30 days. If you are not satisfied, return it for a refund. When you register, we send you source code. Order C-scape today, or call toll free for more information. See why some very major groups are standardizing on C-scape.

C-scape (Lattice/Microsoft/others)

Only \$179

C-scape with Dan Bricklin's Demo Program

\$249

Please add \$3 for shipping; Massachusetts orders add 5% sales tax.

Oakland Group, Inc. 
675 Massachusetts Avenue, Cambridge, MA 02139-3309



CALL TODAY! 617-491-7311 800-233-3733

# If you think you can't afford a UNIX's ystem. we've got a \$160 surprise.

Turn your PC into a multi-user system.

Convert your IBM PC-AT (or compatible) into a multiuser/tasking UNIX work station—at absolutely the best price anywhere, any time. Based on the AT&T-certified UNIX System V/286, the MICROPORT SYSTEM V/AT is designed for use in virtually any computer environment, from office automation to software development.



Runs on virtually all PC-AT clones.

Binary compatible with the AT&T 6300 Plus UNIX System.

Super software-

development

environment



#### Over 200 utilities come standard.

Grep, awk, sort, split, cut, paste, vi and ed (and many more) now let you search and modify files, make use of electronic mail, emulate terminals, calculate electronically, convert data and publish.

SYSTEM V/AT is more than a look-alike. It was derived from AT&T's own UNIX System V release 2 iAPX286. It thereby contains standard System V features the competitors don't support, such as the powerful symbolic debugger, sdb, the shell-layering job-control facility and the F77 Fortran compiler, as well as programming tools such as ctrace, cflow, and bs. Also standard is File System Hardening which greatly reduces data loss in a power

#### Want some more features?

- Console driver providing ANSI terminal interface for monochrome, CGA, Hercules and EGA cards.
- Multiple Virtual consoles allow up to four virtual windows of operation.
- Record and File Locking
- Supports the 286's 16 megabyte virtual address space and fully utilizes its other advance features.
- Supports all standard IBM drive types and most non-standard hard-disk
- Requires only one hard-disk partition. and allows DOS to reside on the same hard disk.
- Provides utilities to transfer files to-andfrom DOS file systems.

We've provided everything: Make, yacc, lex, sccs, cflow, ctrace plus every standard System V software-development tool. The F77 Fortran compiler. And the AT&T Portable C compiler for the 286. Both C and Fortran compilers generate 287 instructions directly—for systems not containing 287 math coprocessors, a kernelresident IEEE-compatible 287 emulator is provided. The large-model code produced by the compiler is among the densest and fastest currently available.

#### So, how do we do it?

MICROPORT offers SYSTEM V/AT at a fraction of the price of the competitors simply because we build on the generic System V/286 product from AT&T. This entire utility package from the certified release has been copied directly to SYSTEM V/AT—without so much as a recompile. Not only does this mean that MICROPORT can offer SYSTEM V/AT at a remarkable low price, it also guarantees a level of quality present in few (if any) other UNIX-system implementations. (And, since our staff was part of the group that implemented the standard System V/286 port for Intel, MICROPORT can offer comprehensive support for the system, as well.)

#### And a dollar change

The price is even better than you thought. Order right away and we'll return one silver dollar just as rapidly, with your product shipment. (If you'd like a little more time we'll apply that dollar to the cost of a brochure—which we'll send right away too.)

NOTE: Educational institutions—Call about our nominally-priced site licensing and source code

60 DAY MONEY-BACK GUARANTEE

ndadolar



To order: Complete the information below. Your attractively-packaged and fully-documented order will be shipped within two weeks.

#### MICROPORT SYSTEMS, INC.

4200 Scotts Valley Drive Scotts Valley, CA 95066 408/438-UNIX or 800/PC2-UNIX (outside CA)

01	107		R A			AT
21	S	E	IVI	V	и	ΑI

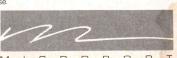
RUNTIME SYSTEM Includes the SYSTEM V/AT op system and over 200 utilities, for two users.	eration
	\$160.00
SOFTWARE DEVELOPMENT SYSTEM The com Software Generation System for 286 development. QUANT:	Jan St. A.
<b>TEXT PREPARATION SYSTEM</b> Includes nroff, troand other programs.	off, spell
QUANT:	\$169.00
THE COMPLETE SYSTEM Contains all three pacindicated above.	kages
QUANT:	\$439.00

□ **OPTIONAL** three to eight-user upgrade. QUANT: \_ Subtotal:

(CA residents add 6.5% tax per copy): Shipping and handling charges (In the USA, \$14.00; in Canada, \$18.00; and in Europe, \$110.

TOTAL DUE:

TELEPHONE\_ ADDRESS\_ STATE USA MASTERCARD BANK DRAFT CHECK CARD NUMBER EXP DATE ☐ Send a brochure only and keep me on your mailing list,



UNIX and DWB are trademarks of AT&T IBM and IBM PC-AT are trademarks of IBM CORPORATION. SYSTEM V/AT is a trademark of MICROPORT SYSTEMS, INC.

Circle no. 154 on reader service card.

#### 68000 MINI FORTH (continued from page 24)

be a flag) from the stack and test its value. If the value is false, execution continues at the word following THEN in the definition: otherwise, execution continues with the word following IF. In the above definition, the phrase 2 MOD simply furnishes the proper flag for IF to "eat." The actions of IF and THEN when ODD is compiled (their compile-time behavior) must ensure that they have the desired effect when ODD is executed (execution-time behavior). Thus the definitions of IF and THEN will define their compile-time behavior, but they must be understood in terms of their execution-time behavior as well. An important point to remember here is that these words, as well as other control words, are not to be invoked directly; they act within definitions to achieve their effects.

Another control word, ELSE, can be used optionally with IF and THEN. If, for example, you wish to redefine the word ODD so that its action is to print the value of an odd number or else drop the number, you might try

#### : ODD DUP 2 MOD IF . ELSE \ THEN ;

The execution-time behavior of *ELSE* is, as you might suspect, to transfer control to the word following *THEN*. A subtle point here is that now *IF* must transfer control to the word following *ELSE* rather than to the one following *THEN* when *ELSE* is present.

FLINT provides two structured loop control mechanisms: DO . . . UNTIL and DO ... WHILE ... LOOP. The execution-time behavior of UNTIL is to eat a flag and pass control back to the word following DO if its value is true. Otherwise the loop completes when control passes to the word following UNTIL. WHILE, on the other hand, makes an exit to the word following LOOP if its value is false. LOOP always passes control to the word following DO. (Note that FLINT's usage of these words is different from standard Forth's; they seem to make a little more sense to me this way.)

Nowhere, I feel, is the elegance and power of Forth's programming paradigm more in evidence than in the definition of the lexicon of words supporting FLINT's structured control. The words ?>, BRA>, BEQ>, and BNE> form the raw material for the tests and branches, and the words MARK, SPLIT, and JOIN provide the tools for assembling them. Each of the control words is then built economically by a single defining word or phrase, and yet when they are used any of the resulting control structures can be nested to arbitrary depth within any of the others!

Forth's elegance and power is evidenced by the lexicon of words supporting FLINT's structured control.

There remains the problem of entering these definitions interactively. In theory this is no worse than entering them into an assembly-language code file-except of course that all is lost when the system is turned off or (as is more likely) crashed by the novice user. My own approach to the problem was to try to find a way to save an image of the augmented system. As it turns out, all that is necessary for saving the state of the system is to preserve the contents of the internal dictionary pointer, register A5, which points to the next vacancy in the dictionary. (This happens to be the area used by TOKEN as temporary storage for the words that it extracts from the line buffer.) The word LOG-OFF accomplishes this task by saving A5 in the system variable BUFPNT, from which it is loaded each time FLINT runs. For this solution to work. you must have the means (via a monitor/debugger, for instance) to save and reload the augmented system in place of the original. Lacking such a tool, you should probably just bite the bullet and build the inner shell words into the source in the manner of the kernel words so that they will be assembled directly in the dictionary. This is tedious work, but it is not difficult if you understand the dictionary structure.

Ultimately, of course, it would be nice to dispense with these primitive methods and be able to create and edit disk-resident files of FLINT source text. In fact, the words SAVE, LOAD, and GO are included in the kernel system to support this very activity. The word GO directs the outer interpreter to take its tokens from a 1K reserved area known as the disk/edit buffer instead of from the line buffer. After the outer interpreter has exhausted the contents of this buffer, it will encounter the token INTERAC-TIVE, which has been placed in memory just beyond the buffer area. Execution of this word reinitiates the normal interactive input sequence and redirects the outer interpreter to take tokens from the line buffer. LOAD and SAVE allow the contents of this area to be read from or written to an absolute disk block called the current block, which is specified as the value of the system variable CBLOCK. A description of these words begs the obvious question of how usable text gets into the buffer (or on disk) in the first place.

My own solution was to use the expanded vocabulary of the inner shell to write a simple editor. Once this editor was built interactively, it could be used to place the text for the inner shell (as well as itself) in the edit buffer for eventual storage on disk via SAVE. These blocks of text are called screens. Careful examination of FLINT's initialization procedure will show that it loads and executes screen #80 (the initialized value of CBLOCK). This block and the succeeding ones are where the text for the inner shell and the editor have been placed. The word -> (defined on screen #80) directs the interpreter to increment CBLOCK and load and execute the next text screen. It allows an entire string of screens to be executed without interactive direction. By this device the inner shell and editor are effectively booted by FLINT whenever it is invoked. The final result is a fairly mature system that contains most of the basic features of a true Forth and enough legitimate tools to be quickly expanded further.

You can characterize the basic outline of FLINT's construction as a bootstrap procedure, which is really just

## We've Got The Missing Link: DDJ Back Issues

#### #78 Volume VIII, Issue 4:

RECLAIM Destroyed Directories—Binary Magic Numbers— 8080 Fig-Forth Directory & File System—SAY" Forth Votrax Driver—TRS-80 8080 to Z80 Translator—Basic Disk I/O, Part II.

#### #81 Volume VIII, Issue 7:

The Augusta Compiler, continued—RED: A Better Screen Editor, Part I—Anatomy of a Digital Vector and Curve Generator— Fast Matrix Operations in Forth, Part II—The AGGHHHI Program—MBOOT Revisited—CP/M Plus Feedback—MS-DOS Rebuttal—68000 Tools—Sizing Memory on the IBM PC.

#### #82 Volume VIII, Issue 8:

Serial-to Parallel: A Flexible Utility Box—McWORDER: A Tiny Text Editor-And Still More Fifth Generation Computers-Specialist Symbols and I/O Benchmarks for CP/M Plus—CP/M Plus Memory Management—Zero Length File Test—PAUSEIF, QUITIF, and now SKIPIF—ACTxx Cross Assemblers.

#### #83 Volume VIII, Issue 9:

FORTH ISSUE: Forth and the Motorola 68000-Nondeterministic Control Words in Forth—A68000 Forth Assembler—GO in Forth—Precompiled Forth Modules—Signed Integer Division-Some Forth Coding Standards—The Forth Sort.

#### #84 Volume VIII, Issue 10:

Unix to CP/M Floppy Disk File Conversion—A Small-C Help Facility—Attaching a Winchester Hard Disk to the S-100 Bus— Using Epson Bit-Plot Graphics—8086/88 Functions Macros— Auto Disk Format Selection—CP/M Plus Device Tables.

#### #85 Volume VIII, Issue 11:

A Kernel for the MC68000—A DML Parser—Towards a More Writable Forth Syntax—Simple Graphics for Printer—Floating-Point Benchmarks.

#### #86 Volume VIII, Issue 12:

Faster Circles for Apples—Cursor Control for Dumb Terminals—Dysan's Digital Diagnostic Diskette—Interfacing a Hard Disk Within a CP/M Environment—The New MS-DOS EXEC Function.

#### #87 Volume IX, Issue 1:

A structured Preprocessor for MBASIC—A Simple Window Package—Forth to PC-DOS Interface—Sorted Diskette Directory Listing for the IBM PC-Emulate WordStar on TOPS-20-More on optimizing compilers—The PIP mystery device contest.

#### #88 Volume IX, Issue 2:

Telecommunications Issue: Micro To Mainframe Connection— Communications Protocols—Unix to Unix Network Utilities— VPC: A Virtual Personal Computer for Networks—PABX on the Personal Computer—BASIC Language Telecommunications Programming—U.S. Robotics S-100 Card Modem.

#### #89 Volume IX, Issue 3:

RSA: A Public Key Cryptography System, Part I—Introduction to PL/C: Programming Language for Compilers—Program Design Using Pseudocode—More on Binary Magic Numbers—How fast is CP/M Plus?—CP/M 2.2 BIOS Function SELDSK—The results of the Floating-Point benchmark.

#### #90 Volume IX, Issue 4:

Optimizing Strings in C-Expert Systems and the Weather-RSA: A Public Key Cryptography System, Part II—Several items on CP/M Plus, CP/M 2.2 Compatibility—BDOS Function 10: Vastly improved—More on MS-DOS EXEC Function—Low-Level Input-Output in C.

#### #91 Volume IX, Issue 5:

Introduction to Modula-2 for Pascal Programmers—Converting Fig-Forth to Forth-83—Sixth Generation Computers—A New Library for Small-C-Solutions to Quirks in dBASE II.

#### #92 Volume IX, Issue 6:

CP/M on the Commodore 64—dBASE II Programming Techniques—First Chinese Forth: A Double-Headed Approach—cc-A Driver for a Small-C Programming System—A New Library for Small-C (Part II)—Comments on Sixth Generation Computers—Review of Turbo Pascal.

#### #95 Volume IX, Issue 9:

Forth Special Issue—File Maintenance in Forth—Forth and the Fast Fourier Transform—Computing with Streams—A Forth Native-Code Cross Compiler for the MC68000—The FVG Standard Floating-Point Extension—CP/M Plus Interbank Memory Moves Without DMA-Ways to make C more powerful and flexible

#### #96 Volume IX, Issue 10:

More dBASE II Programming Techniques—Simple Calculations with Complex Numbers—GREP.C: A Unix-like Generalized Regular Expression Parser—An optimization scheme for compilers, MSDOS 2.0 Filters, Sizing RAM under MSDOS, Two programming systems illustrating Runge-Kutta intergration.

#### #97 Volume IX, Issue 11:

Adding Primitive I/O Functions to MuLISP—Program Monitor Package: Using Interrupts to Instrument Applications—CP/M 2.2 Goes PUBlic—A Guide to Resources for the C Programmer— **RESORT** 

#### #104 Volume X, Issue 6:

Information Age Issues—Modems: 2400 Bit/Sec and Beyond— C UART Controller—Christensen Protocols in C.

#### #108 Volume X, Issue 10:

Special Forth Issue—A Threaded-Code Microprocessor Bursts Forth—Design a Forth Target Compiler.

#### #109 Volume X, Issue 11:

Modula-2 vs. Pascal for Microcomputers: An Update-Bit Manipulation in Modula-2.

#### #113 Volume XI, Issue 3:

Parallel Processing—Concurrency and Turbo Pascal—What Makes DOS Fast—Minimizing Arbitrary Functions—MC68000 vs. NS32000.

#### #114 Volume XI, Issue 4:

Special Al Issue—Programming in LISP and Prolog—An Expert at Life—Perils of Protected Mode—I/O Redirection for the

#### #115 Volume XI, Issue 5:

Software Design from the Outside In-Dan Bricklin's DEMO Program—Cryptographer's Toolbox—EGA Graphics & Fast PC Graphics—How to Write Memory Resident Code.

#### #116 Volume XI, Issue 6:

Telecommunications Without Errors—General-Purpose Sorting-Structured Programming.

#### #117 Volume XI, Issue 7:

Special FORTH Issue Forth Standards Proposal—Forth in a Bottle—Forth & Expanded Memory—Forth Structured Program-

#### #118 Volume XI, Issue 8

Special C Issue Benchmarking C Compilers—The Joy of Conciseness—Nearly Perfect Trees—Generics in Ada—Real-World Data Types

-	-000	-	months.	CONTRACTOR OF THE PERSON.	Chillian	STATE OF THE PARTY.
70						ECD 4
- E 1			SPEC.			

Return this coupon with payment to Dr. Dobb's Journal, 501 Galveston Dr., Redw

Please send me the issue(s) circled: 80 81 82 83 84 85 87 88 89 90 91 92 95 104 108 109 97 110 113 114 115 116 117 Price: 1 issue—\$5.00. 2-5 issues—\$4.50 each. 6 or more—\$4.00 each. (There is a \$10.00 minimum for charge orders.)

I have read the postal instructions and understand that I will not receive my order unless I have sent the correct payment amount.

Please charge my:	Visa	□ M/C	Amer. Exp.	
Card No.			Exp. Date	

Signature \_

Outside the U.S., add	\$.50 per issue.	noney order).	
Name			
Address			
City	1		
State		Zip	

Outside U.S., add \$.50 per issue. Price includes shipment by second class or foreign surface mail. Within U.S., allow 9 weeks for delivery. For U.P.S. shipment, add \$1.00 for 1-2 issues and \$.50 per issue thereafter—NO P.O. BOXES. Airmail rates: Canada—add \$1.75 per issue; other foreign add \$3.00 per issue.

Please provide a street address rather than a P.O. Box.

### Dr. Dobb's 1986 Listings on Disk

Dr. Dobb's Journal of Software Tools offers the convenience of selected listings on disk.

#### DR. DOBB'S LISTINGS # 1

#### JANUARY—APRIL 1986

Includes listings from the following articles, and more.

January Issue #111

"A Simple OS for Realtime Applications; 68000 Assembly Language Techniques for an Operating System Kernel" by DDJ editor Nick Turner.

"32-bit Square Roots; An 8086 Assembly Language Routine for 32-bit Square Roots" by Michael Barr.

#### February Issue #112

"Data Abstraction with Modula-2" by Bill Walker and Stephen Alexander.

"Learning Ada on a Micro; A Draw Poker Program in Ada" by DoWhile Jones.

#### March Issue #113

"A Variable-Metric Minimizer; A C Program for Minimizing Arbitrary Functions" by Joe Marasco.

"Concurrency and Turbo Pascal; An Approach to Implementing Coroutines in Pascal" by Ernest Bergmann. **April Issue #114** 

"Boca Raton Inference Engine; Lisp, Prolog, and Expert-2 Techniques and Code" by Robert Brown.

Dr. Dobb's Listings #1/86

Item #170 \$14.95

#### DR. DOBB'S LISTINGS #2

#### MAY-AUGUST 1986

Includes listings from the following articles, and more. May Issue #115

"Simple Plots with the Enhanced Graphics Adapter" by Nabajyoti Barkakati.

"The Cryptographer's Toolbox" by Fred A. Scacchitti. June Issue #116

"Structured Programming; Overloading Procedures, Exporting Opaque Types, Data Hiding" by Namir Shammas. "Compuserve B Protocol" by Steve Wilhite.

#### July Issue #117

"Structured Programming; Tiny Tools, Array-Defining Words" by Michael Ham.

#### August Issue #118

"Structured Programming; Generic Routines in Ada and Modula-2, Extended For Loop" by Namir Shammas.

Dr. Dobb's Listings #2/86

Item #171

#### OBB'S LISTINGS #3

#### SEPTEMBER—DECEMBER 1986

Includes listing from the following articles, and more:

#### September Issue #119

"Algorithms: Curve Fitting with Cubic Splines" by Ian E. Ashdown.

"C Chest: Directory Traversal, Trailing Zs, and Horrifying Experiences" by Allen Holub.

#### October Issue #120

"C Chest: More, a File-Browsing Utility" by Allen Holub. "Processors: TNZ: An 8-bit to 16-bit Translator" by Richard Campbell.

#### November Issue #121

"Digital Dissolve" by Mike Morton. "Mandelbrot Set" by Howard Katz.

#### December Issue #122

"In Search of a Sine" by R.A. Campbell.

"16-bit Software Toolbox: A demo program to illustrate the undocumented MS-DOS Int 2EH route to the command interpreter in command.com" by Ray Duncan and his readers, and 8086-68000 based string comparison routines.

Dr. Dobb's Listings #3/86

Item #172 \$14.95

Please specify MS-DOS, Macintosh, or CP/M. For CP/M disks specify: Apple, Osborne, Kaypro, Zenith Z-100 DS/DD, 8" SS/SD (listings no. 1, 2, and 3 only)

#### JANUARY 1987 LISTINGS

Now you can also receive on disk all the source code for articles in this issue! Available formats: MS-DOS, Macintosh, Kaypro

Dr. Dobb's Listings #1/87

City

Item #17187 \$14.95

Please send me:	Listings Disk #1/86 for \$14.95
	Listings Disk #2/86 for \$14.95
	Listings Disk #3/86 for \$14.95
	Listings Disk #1/87 for \$14.95
CA	residents add sales tax %
	Add \$1.50 per item for shipping
	TOTAL

Specify Format  MS-DOS	☐ Macintosh ☐ CP/M ☐ Apple ☐ Osborne ☐ 8"SS/SD ☐ Kaypro ☐ Zenith
Check encl.	Charge my: VISA M/C AmerEx
Card #	Exp. Date
Signature	
Name	
Address	

State \_

Zip\_

an accelerated and miniaturized version of the conventional process whereby low-level tools are used to build higher-level ones. The assembler is used to build an interactive interpreter/compiler (the FLINT kernel), which is used to incorporate a more sophisticated set of tools (the inner shell). These are then used to build an editor that allows the inner shell, the editor, and all future extensions to become permanent parts of the system.

#### The Aftermath

What, you might ask, did the teacher learn from all this? Well, for one thing. I relearned the value of ignorance. I sailed pretty far into the project on the momentum of my initial enthusiasm. It lasted, in fact, until everything was pretty much complete and working. Unfortunately, the job is not done when the program works. Cleaning up little messes, pruning, tuning, documenting, testing, and tracking down the subtle bugs—in short, the real work—was equally time-consuming but much less rewarding. Determination and good old-fashioned stubbornness had to finish what enthusiasm had begun. If I had fully realized in advance the amount of extra work involved, I would probably never have started. The moral here is that underestimating the magnitude of a task is often a necessary condition for attempting it. The work is still not finished, of course. It never is. The stack is in the wrong place, the realization of the CODE submode is imperfect, and so on, and so on. Any program is only an approximation of what it should be.

As far as Forth itself is concerned. there are several things to be said. I have acquired a terrific amount of respect for the ingenuity of Forth's inventor, Charles H. Moore. Work of genius is a term properly applied to such an engagingly simple yet immensely practical conception. I have not yet become, however, one of Forth's true believers. Programming is not my profession—it's actually more of a hobby-so I don't have enough real knowledge or experience to make any credible claims for its superiority or inferiority to other programming languages. I'll leave that issue to the people who make their money writing programs. My

opinion is that Forth's programming paradigm of building the language to fit the problem allows (and indeed requires) a much more flexible and imaginative approach to programming problems than is called for with many more traditional lan-

The extra degree of freedom actually seems to require more self-discipline on the part of the programmer to use it wisely, but it makes problem solving more fun. I find that my programs tend to become compositions that I judge on aesthetic as well as functional grounds.

#### Availability

All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to Dr. Dobb's Journal, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number and disk format (MS-DOS, Macintosh, Kavpro).

DDJ

(Listings begin on page 52.)

Vote for your favorite feature/article. Circle Reader Service No. 3.

# **New from Lifeboat:**

#### ADVANTAGE C++

#### Brings the power of C++ to your PC.

- Opens the door to object-oriented programming
- Allows programs with greater resilience, fewer bugs
- · Fully compatible with existing C programs
- · All the benefits of C without its limitations

#### **ADVANTAGE Link**

#### Everything you've always wanted in a PC-DOS linker.

- The fastest, most powerful PC-DOS linker available
- The first linker to take full advantage of extended memory
- Accepts Microsoft and Phoenix command files
- Supports up to 53 commands—more than any other linker
- Compatible with Microsoft CodeView

#### ADVANTAGE LIBRARY SERIES

#### **TimeSlicer**

#### Multi-tasking library streamlines C programming.

- Perform concurrent tasks and real-time event processing
- · Includes header files for both C and assembly language and example programs with source code
- Compatible with C++ and object-oriented programming
- Critical resource management assured

To order or to obtain complete specification sheets, call: 1-800-847-7078 In NY: 914-332-1875 55 South Broadway Tarrytown, NY 10591



The Full-Service Source for Programming Software.

Circle no. 118 on reader service card.

# The OS-9 Operating System

by Brian Capouch

he OS-9 operating system is a modular, multiprogramming, multitasking operating system (OS) that runs on a large variety of Motorola microprocessors. Designed to be conceptually similar to Unix, it provides programmers with a good environment for the same reasons that Unix does: it uses a hierarchical file structure, allows command-line invocation of concurrent processes, has a similar implementation of pipes and filters, and permits device-independent I/O. The differences are more important than the similarities, though.

#### Mean and Lean

OS-9 differs from Unix in several significant respects. It is, to turn a phrase, "meaner and leaner" than Unix. It occupies 12K of address space in its smallest 6809 incarnations and a little more than 48K in its full-blown 68020 form. It is more dynamically changeable than Unix, allowing users to add I/O devices and update system modules without rebooting a running system. It was also designed specifically to accommodate ROMed software easily and, for a variety of reasons, has proven suitable for real-time and process-control applications. These and other features have assured OS-9 a niche encompassing a wide variety of markets. This article gives an overview of the design of OS-9, particularly focusing on those aspects that make it special. I've included a pair of small application programs that are intended to give you the flavor of the OS-9

Brian Capouch, R. R. 2 Box 151, Monon, IN 47959-9229. Brian is a self-taught computer scientist who teaches at St. Joseph's College in Rensselaer, Ind.

#### OS-9 is meaner and leaner than Unix.

programming environment

#### **Origins**

OS-9 originally had its genesis as part of a contract project between its designer, Microware Systems Corp., and Motorola, during the time that the hardware design of the 6809 processor was being finalized. The original concept of the project was to provide a modern, structured version of the BASIC programming language that would take advantage of the features of the 6809. The resultant language, called Basic09, incorporates several features that were at the forefront of language design; I'll review it briefly with examples later.

As the language development proceeded, Microware, with an eye toward the developing academic and commercial use of Unix, began developing an operating environment to complement Basic09's structure and modularity. Thus OS-9 was born. The language and the operating system appeared together in 1981, a few months after the 6809 came into production. As the design for the 68000 was begun shortly thereafter, Microware began a port of OS-9 for that processor.

As the 68000 and its successors have appeared in a multitude of applications, OS-9 has been ported into hundreds of designs. OS-9 currently appears on a wide range of machines, ranging from the 6809-based Tandy

Color Computer to the GMX Micro-20, a 68020-based system that provides 2-megabytes of RAM, SASI and floppy-disk interfaces, serial and parallel I/O ports, and a 68881 math coprocessor on a card that mounts on a 5½-inch disk drive. This machine has a performance that outstrips a variety of machines, including the VAX 11/780.

#### Named Modules

Structurally speaking, perhaps the most significant feature of OS-9 is its extreme level of modularity (see Figure 1, page 31). The underlying model of the OS-9 address space is a dynamic collection of named modules. When an executable module comes to life as a process, it is associated with a separate area of memory that is used to store its data. All code exists in what are known as memory modules, whether located in primary or secondary store. A memory module, generally, is a segment of code sandwiched between a header and a cvclic rendundancy checksum. The header contains information about the module that is used by the OS both during the time that the module is resident in memory and during transfers to and from secondary storage devices. The checksum is used to verify the integrity of the module during transfers to and from secondary store.

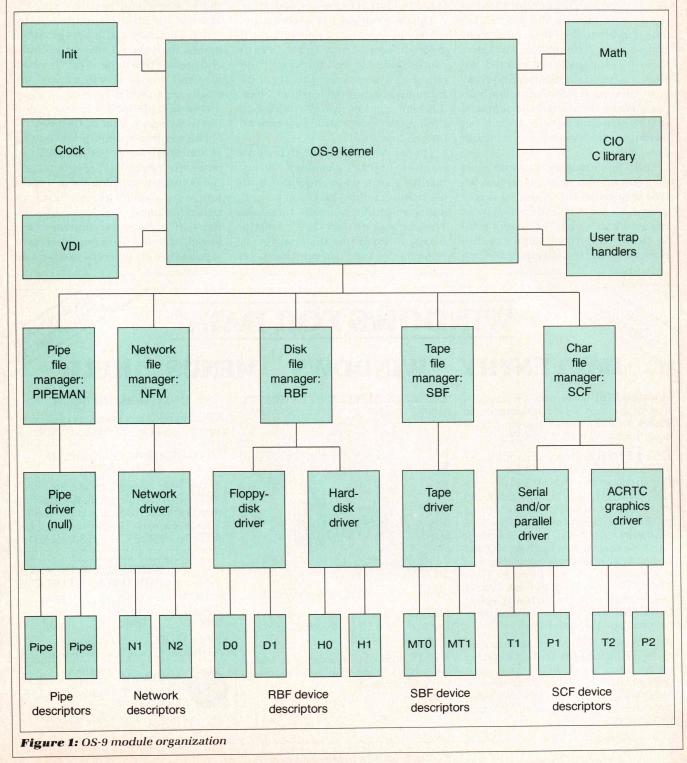
Modules that contain object code must consist of position-independent code and are shared automatically between various users of the system. This avoids having copies of the same object code occupying multiple sections of memory. In such cases, the OS assigns each user of the module to a different area of memory for data storage.

The first bytes of each module are "sync bytes." These bytes, which consist of unused machine opcodes, are used at start-up to locate and load ROMed modules automatically. OS-9 permits ROMed code to be updated dynamically, simply by loading into memory a module with the same name as the ROMed module but with a higher revision number in its header. The mixture of modules that con-

stitutes the "current environment" is more dynamic than in most other OSs, a feature that has opened up many application avenues.

#### Layered Hierarchy

Like Unix, OS-9 is organized as a layered hierarchy of functional modules. Each component is a memory module, as described earlier. At the lowest level of the hierarchy is the kernel, aided by small init and clock modules as well as several others. The OS-9 kernel handles all basic OS functions, including process scheduling and dispatch, memory management, and basic I/O processing. It determines from the init module the exact characteristics of its environment, which under OS-9 more than under Unix is likely to differ from one system to another. The clock



OS-9 OPERATING SYSTEM (continued from page 31)

module interfaces to a tick generator for time-slice timing.

At the next layer outward from the basic system modules lie the file manager modules. Each of these is designed to interface the I/O data stream into and out of the processor and a particular class of similar devices. Because the conditioning is performed outside the kernel, all data appears to the CPU as equivalent byte streams. The module RBF interfaces to random-block-oriented devices, such as disks. SCF handles character streams that are bound for both parallel and serial I/O devices, and PI-PEMAN coordinates data streams between concurrently processes.

Each file manager conditions its data stream and passes it on to a device driver module. A device driver must exist for each different type of hardware to which the system is interfaced. The driver, such as a disk controller or intelligent interface processor, may control a large number of devices, and it is this level of the system that hides hardware dependencies from the lower levels of the OS.

At the highest level of abstraction are the device descriptor modules. The system needs one of these for each individual I/O device. They contain device-dependent parameters, such as port addresses, disk blocking factors, control characters, and so on.

The OS-9 unified I/O system is dynamically configurable, making it quite different from Unix. Devices can be added to and removed from running systems on the fly, and through changing device descriptors, different devices can be added to the same port and referenced interchangeably.

The 68xxx versions of OS-9 contain a complete math subroutines library that handles a wide variety of floating-point, extended-precision integer, and transcendental math functions. This package is called via the *Trap* instruction. Programs that use the library can be used without change on

systems that implement hardware coprocessors by changing the trap handler associated with the call.

#### The User Interface

The OS-9 user communicates with the OS through a user interface called shell. Shell is similar to Unix shells, although it differs in enough respects to wreak havoc with the neuronal patterns of folks who try to time share between the two different OSs. Many of the utility commands go by what Microware thinks are saner names than their Unix equivalents, and utilities that depend on file structure and process specifics are of necessity organized differently.

One notable difference between the two shells is a set of OS-9 control characters that speed up line editing at the shell level. For instance, in other environments I have found myself constantly wishing for a "repeat line" control character (commonly assigned to Control-A) that provides the function "reenter the contents of the line input buffer up to the most recent carriage return entered." High-speed

FOR CHIN

#### WINDOWS FOR DATA<sup>TM</sup>

#### WINDOWS MENUS HEL

**Windows for Data** does the hard jobs that others can't — we **guarantee** it. Makes standard display and entry tasks easy. Reliable. Compact. Portable.

**DATA ENTRY** 

DATA ENTRY: The most complete and flexible data entry system on the market. Pop-up data-entry windows; field types for all C data types, plus decimals, dates, and times; auto conversion to and from strings for all field types; system and user-supplied validation functions; range-checking; scrollable context-sensitive help; required and must-fill fields; programmer-definable edit keys, field types, and field masks. Read field by field or auto-read all fields. Branch and nest window forms. Virtually every capability of WFD can be modified to meet special needs.

WINDOWS: WFD is built upon and includes Windows for C, the windowing system rated #1 in PC Tech Journal (William Hunt, July 1985). WFC now has more features than ever, including automatic full compatibility with Microsoft Windows and TopView.

**UNPRECEDENTED FLEXIBILITY** 



As many possibilities as Vermont in June.

**MENUS:** Build multi-level menus in the format of Lotus 1-2-3, Macintosh, or a style of your own choosing.

**HELP:** Build context-sensitive or menudriven help systems. Display text in pop-up, scrollable windows

#### UNIX, DOS, OR BOTH

WFC and WFD provide source code compatibility between PCDOS and UNIX.

#### OUR CHALLENGE AND GUARANTEE

If you have an application where no other tool can do the job, try **Windows for Data.** If it doesn't help you solve your problem, RETURN FOR A FULL REFUND. YOU MUST BE SATISFIED.

#### WINDOWS FOR DATA FOR C

PC DOS\* \$295 \$195 XENIX-286 \$595 \$395 UNIX CALL CALL

Call for **FREE Demo diskette**.
\*All popular C compilers; no royalties.



21 Elm Ave. Richford, VT 05476 **802-848-7738**, ext. **31**.

MasterCard & Visa Accepted. Shipping \$3.50 VT residents add 4% tax.



#### Programmer's Paradise Gives You Superb Selection, Personal Service and Unbeatable Prices!

Welcome to Paradise. The PC/MS-DOS software source that caters to your individual programming needs. Discover the Many Advantages of Paradise...

- Lowest price guaranteed
- Latest versions
- Huge inventory
- Immediate shipment
- Special orders
- 30-day money-back guarantee

#### Ve'll Match Any Nationally Advertised Price

	LIST OURS		LIST OU	JRS		LIST (	OURS
C++ ADVANTAGE C++ PFORCE++	\$ 495 CALL 395 CALL	C UTILITY LIBRARIES ASYNC MANAGER BASIC C C ESSENTIALS		135 129 85	SORT UTILITIES AUTOSORT M/SORT OPT-TECH SORT	150 155 149	129 139 115
C COMPILERS C-86 PLUS DATALIGHT - C DATALIGHT - C DEVELOPER'S KIT	497 CALL 60 49 99 79 500 289	C FOOD SMORGASBORD W/SOURCE C TOOLS PLUS ENTELEKON COMBO PACKAGE C FUNCTIONS LIBRARY	150 300 175 200	98 188 135 169 109	MAKE, LINT, PROFILE, UTILITIE C CROSS REFERENCE GENERATOR LMK POLYMAKE		39 145 78
LATTICE C 3.2 LATTIC C W/SOURCE LET'S C W/CSD DEBUGGER	900 545 75 59 150 109	C WINDOWS SUPERFONTS FOR C ESSENTIAL C UTILITY LIBRARY ESSENTIAL COMM LIBRARY	130 50 185	109 43 135 135	OTHER POLYTRON PRODUCTS PMAKER PFINISH	CALL (125 395	CALL 95 245
MICROSOFT C 4.0 MARK WILLIAMS C SUPERSOFT C WIZARD C	450 285 495 289 395 339 450 369	W/BREAKOUT DEBUGGER GREENLEAF FUNCTIONS GREENLEAF COMM THE HAMMER	250 185 185	195 135 135 175	THE PROFILER PC LINT PRE-C TEXT MANAGEMENT UTILITIES	125 139 295 120	94 105 165 94
C INTERPRETERS C-TERP INSTANT C	300 235 500 379	MULTI C PFORCE TIMESLICER TOPVIEW TOOLBASKET	149 395 295 250	135 245 265 189	DEBUGGERS ADVANCED TRACE 86 BREAKOUT	175 125	139
INTRODUCING C RUN/C RUN/C PROFESSIONAL 1.1	125 105 150 89 250 169	SCREEN DISPLAY, WINDOWS C WORTHY	295	269 94	CODESMITH 86 C SPRITE CI PROBE	145 175 75 75	108 138 59 59
ASSEMBLERS, LINKERS 386IASM ADVANTAGE LINK MACRO-86	495 395 495 CALL 150 98	CURSES W/SOURCE FLASH UP WINDOWS MICROSOFT WINDOWS	125 250 75	184 68	CSD SOURCE DEBUGGER PERISCOPE I PERISCOPE II PERISCOPE II-X	295 145 115	249 109 85 245
PASM-86 PLINK 86 PLUS QUELO 68000 X-ASM	195 135 495 335 595 509	DEVELOPMENT SYSTEM ON-LINE HELP PANEL SCREENPLAY (LATTICE)	500 149 295 150	329 109 224 135	PFIX 86 PLUS XVIEW 86	395 60	49
January Special From Phoenix	ls	SOFTSCREEN HELP VIEW MANAGER VITAMIN C VC SCREEN	195 275 150 99	175 199 135 84	Featured Produ		
PASM-86 PDISK PFANTASY PACK PFINISH PFIX PLUS	195 135 195 135 1295 889 395 245 395 245	WINDOWS FOR C WINDOWS FOR DATA Z VIEW FILE MANAGEMENT	195 295 245	145 250 189	ADVANTAGE LINK—the first over take advantage of extended memory. I modules from Microsoft languages, RF Fortran. Lattice C, and many more. St memory caching, object file merging, or	Link object M COBOI apports complex	ct L and
PFORCE PLINK 86 PLUS PMAKER PMATE PRE-C	395 245 495 335 125 95 195 125 295 165	BTRIEVE XTRIEVE WREPORT GENERATION BTRIEVE/N XTRIEVE/N WREPORT GENERATION CTRIER	245 245 390 595 595 940	195 315 465 465 750	overlay structures and automatic over	lay reload Special	ling. \$349

EDITORS BRIEF	195 CA	LL
CVUE	75	59
W/SOURCE	250	195
EDIX	195	155
EMACS	295	265
EPSILON	195	159
FIRSTIME (C)	295	229
KEDIT	125	105
LSE	125	95
PMATE	195	125
PC/Vi	149	129
SPF/PC	195	149
VEDIT	150	109
VEDIT PLUS	225	139
VEDITIEUS		

Programmer's Paradise

## TOOLS FOR TURBO PASCAL ALICE FIRSTIME FLASH UP WINDOWS HALO SCREENPLAY SCREEN SCULPTOR T-DEBUG PLUS TURBO EXTENDS TURBO PASCAL ASYNC MGR TURBO PROFESSIONAL TURBO POWER TOOLS PLUS TURBO WINDOWS **NEW Products**

PASCAL COMPILERS
MICROSOFT PASCAL PASCAL 2 TURBO PASCAL

OTHER BORLAND PRODUCTS

LIST OURS

CALL CALL

395 100 355

386IASM/LINK — Complete development package for 80386 microprocessor including an assembler, linker, and debugger. Upwardly compatible with Microsoft's Macro Assembler.

List \$495 Ours CALI

LATTICE C — Version 3.2 — Features full support for Microsoft Windows including the "Far," "Near," and "Pascal" key words.

List \$500 Ours \$28

PASCAL 2—Highly optimized Pascal compiler, with source level debugger, profiler.
List \$395 Ours \$

PFORCE + + — Huge library of functions designed specifically for object-oriented programming with C + +.
List \$395 Ours CALL

RUN/C PROFESSIONAL — Version 1.1 — Now compatible with Microsoft 4.0! Loadable libraries advanced debugging features.

List \$250 Ours \$10

TIMESLICER—Multitasking, linkable library supporting concurrent tasks and real-time event processing with header files provided for both C and Assembly.

List \$295

Ours \$26 Ours \$265

BASIC		
BETTERBASIC	199	139
SUMMIT ADD ONS	CALL (	
BETTER TOOLS	95	89
FINALLY	99	89
MICROSOFT QUICKBASIC	99	75
PROFESSIONAL BASIC	99	75
8087 MATH SUPPORT	50	45
PANEL-BASIC	145	115
RM/BASIC	600	479
TRUE BASIC	150	105
OTHER PRODUCTS AVAILABLE TO	THE BA	SIC
PROGRAMMER INCLUDE MULTIH		
BTRIEVE, GSS GRAPHICS, SCREE	N SCULPT	OR.
STRUBAS, 87 BASIC.		
STROBIES, OF BROICE.		

COBOL COMPILERS/UTILITIES		
MICROSOFT COBOL	700	445
MICROSOFT COBOL TOOLS	350	205
MICROSOFT SORT	195	139
MICRO/SPF	175	CALL
OPT-TECH SORT	149	115
REALIA COBOL	995	785
SCREENPLAY	175	155
RM/COBOL	950	639
RM/COBOL 8X	1250	895
VISUAL COBOL (MBP)	1150	1015
FORTRAN UTILITIES		
ACS TIMES SERIES	495	419
nes inde obities	0.00	000

87 SFL	250	22
FOR-WINDS	90	78
FORLIB-PLUS	70	54
GRAFMATICS OR PLOTMATICS	135	119
GRAFMATICS AND PLOTMATICS	240	219
FORTRAN SCIENTIFIC		
SUBROUTINES	295	249
POLYFORTRAN TOOLS I	179	143
STRINGS AND THINGS	70	54
ALSO AVAILABLE TO THE FORTRAN	1	
PROGRAMMER: PANEL, MULTIHALO	O, BTRI	EVE,
ESSENTIAL GRAPHICS, FLASH UP W	INDOW	S,
GSS GRAPHICS, OPT-TECH SORT.		
GDD GIGH IMEE, GT		

PROLOG ARITY PROLOG (STANDARD) ADDIT. ARITY PRODUCTS CHALCEDONY PROLOG TURBO PROLOG LISP, OTHER AI, CALL FOR INFORMATION, PRICING, AVAILABILITY.	CALL C 100 100	59 CALL 89 79
TRANSLATORS/BRIDGES BASTOC (MBASIC) C TO DBASE DBC III	495 150 250	399 135 189

BASTOC (MBASIC)	495	399
TO DBASE	150	135
OBC III	250	189
W/SOURCE	500	378
BX	350	329
FORTRIX	6000	CALL
R: BRIDGE	395	319

#### **Terms and Policies**

GRAPHICS
ESSENTIAL GRAPHICS
GSS GRAPHICS DEVELOPMENT
TOOLKIT
GSS KERNEL SYSTEM
GSS METAFILE INTERPRETER
GSS PLOTTING SYSTEM
HALO—ONE LANGUAGE
HALO—FIVE MICROSOFT
LANGUAGES
METAWINDOWS
METAWINDOWS PLUS
METAFONTS
METAFONTS
METAFONTS PLUS

We honor MC, VISA, AMERICAN EXPRESS
No surcharge on credit card or C.O.D. Prepayment by check. New York State residents add applicable sales tax. Shipping and handling \$3.00 per item, sent UPS ground. Rush service available, prevailing rates.

C TREE R TREE

W/SOURCE DB VISTA
W/SOURCE
DB QUERY
W/SOURCE

FABS FABS PLUS INFORMIX INFORMIX 4GL INFORMIX SQL

PHACT

CQL DBC III

205 250

- Programmer's Paradise will match any current nationally advertised price for the products listed in this ad.
- Mention this ad when ordering—some items are specially priced.
   Prices and Policies subject to change without notice.
   Corporate and Dealer inquiries welcome.

-800-445-7899 In NY: 1-800-642-6471

> 487 E. Main Street, Mt. Kisco, NY 10549 914-332-4548

Programmer's

Circle no. 334 on reader service card.



#### OS-9 OPERATING SYSTEM (continued from page 32)

interaction with any processor leads to a large number of typographical mistakes, such as misspelling directory, file, or command names. The repeat-line character allows for quick reentry of the offending line, then backspacing to the point of error without having to retype the preceding characters. Although this may seem a picayune advantage to some, it can become addictive. Other control characters allow canceling the line, pausing the screen display, interrupting and canceling the currently running program, and so on. The characters that these functions map to are defined in the device descriptor and can be changed with the utility programs tmode and xmode. OS-9 also supports Unix-style type-ahead on serial input devices.

#### **Development Tools**

OS-9 programmers can choose from a variety of programming languages and functional processors. The OS-9 C compiler is Unix compatible down to the standard library level, and most of the system calls accessible from C have been assigned names that correspond to their Unix equivalents. One OS-9 hacker reported developing a reasonably complex program that included several low-level I/O calls using OS-9 C on his Radio Shack Color Computer. He then ported the program to a VAX 11/780, where it compiled and ran without a single change.

Besides C, there also exist compilers

for Pascal, FORTRAN-77, COBOL, and at least four different versions of the BA-SIC language. These include Basic09, the language for which OS-9 was originally developed. Basic09 is an interesting and unusual language, and the example programs presented later demonstrate some of its features. Besides language processors, there is the usual plethora of functional processors, such as text editors, formatters, and so on. The standard OS-9 assembler is a relocatable macro assembler that allows management of complex assembly-language code in a variety of libraries that can be assembled separately.

#### **More Differences**

A few sundry points also underscore the differences between OS-9 and

```
Microware OS-9/68000 Resident Macro Assembler V1.6 86/08/18 14:02 Page
 syscall.a
OS-9 Assembly constants -
00001 *! Syscall Routine - From Microware Manual
00002
                            use
                                      <oskdefs.d>
00001
00072
00003
00004
       00000000
                            org
                                      0
00005
       00000000 Return
                            do.1
                                      1
00006
       00000004 Length1
                            do.1
                                      1
00007
       00000008 Param2
                            do.1
                                      1
80000
       0000000c Length2
                            do.1
00009
00010
                                      SysCall, (Sbrtn<<8) !Objct, (ReEnt<<8) !1,0,0, SysCall
                            psect
00011
00012 0000 0c80 SysCall
                            cmpi.1
                                      #2, d0
                                                     check parameter count
00013 0006 6640
                            bne.s
                                      ParamErr
                                                     branch if error
00014 0008 Ocaf
                                      #4, Length1 (a7) is first parameter integer?
                            cmpi.1
00015 0010 6636
                                                     branch if not
                            bne.s
                                      ParamErr
00016 0012 0caf
                            cmpi.1
                                      #52, Length2(a7) 52 bytes of registers?
00017 001a 652c
                            blo.s
                                      ParamErr
                                                     branch if not
00018 *
                Now put model on the stack
00019 001c 343c
                            move.w
                                      #Modllen/2,d2 number of words for dbra
00020 0020 41fa
                                      Model+Modllen(pc), a0 get address of model
                            lea
00021 0024 6002
                                      SysCO2
                            bra.s
                                                     branch into loop
00022 0026 3f20 SysCO1
                                      -(a0), -(a7)
                            move.w
                                                     move a word
00023 0028 51ca SysCO2
                            dbra
                                      d2, SysCO1
                                                     continue if not done
00024 002c 2041
                            move.1
                                      d1, a0
                                                     point to function code
00025 002e 3f68
                            move.w
                                      2(a0),2(a7)
                                                     set function code
00026 *
                    Get the registers
00027 0034 2a6f
                                     Param2+Modllen(a7),a5 get address of parameter
                            movea.1
00028 0038 4cdd
                            movem.1
                                      (a5)+, d0-d7/a0-a4 get register
00029 003c 4e97
                            isr
                                      (a7)
                                                     call function
00030 003e 48e5
                                      d0-d7/a0-a4,-(a5) copy register set
                            movem.1
00031 0042 4fef
                            lea.1
                                      Modllen(a7), a7 clear stack
00032 0046 4e75
                            rts
00033
00034 0048=323c ParamErr
                            move.w
                                      #E$Param, d1
                                                     get error code
00035 004c-003c
                                      #Carry,ccr
                                                     set carry
00036 0050 4e75
                            rts
00037
00038 0052=4e40 Model
                            os9
                                      F$Fork
                                                     model system call to put on stack
00039 0056 4e75
                            rts
00040
00041
       00000006 Modllen
                            eau
                                      *-Model
00042
       00000058
                            ends
00043
Errors: 00000
Memory used: 19k
Elapsed time: 2 second(s)
```

Code Example 1: Syscall, an OS-9 resident macro assembler

Unix. First, the OS-9 kernel is written in assembly language instead of in C. Although this restricts the OS to a small set of machines, it results in faster, more compact code. OS-9 also uses a different scheduling algorithm—one that results in noticeably faster throughput than most Unix implementations. Because of its modular memory management and dynamic configurability, OS-9 lends itself better to low-level control applications and real-time processing. On the other hand, OS-9 does not swap programs into and out of primary store. Although this speeds up throughput considerably, it also means that once the total available RAM in a system is occupied, no more jobs can be run. OS-9 has no limit on the number of concurrently executing processes—one user reports spawning more than 600 of them on his 68020 system before running out of memory. OS-9 users also report that the substantially smaller memory requirements and faster speed of OS-9 suit it for many applications in which Unix is too large and slow.

#### Basic<sub>09</sub>

The example programs I have included with this article are designed to illustrate some interesting facets of OS-9. The Basic09 programming language is a highly modular implementation of BASIC that is not only compatible with most sophisticated versions of that language but also contains many structures that are seen in Pascal. It is an interpretive compiler that attempts to balance the better points of both translation processes. The language contains both an integral editor and debugger. Source code is compiled, a line at a time as it is entered, into an intermediate code. Syntax errors are reported to the user as each line is entered and can be immediately corrected using the editor. Programs consist of any number of named procedures in which a variety of data types can be declared and built up into named Pascal-recordlike structures. Parameters can be passed between procedures by either value or reference, and TRACE, PAUSE, and STATE statements allow for interactive debugging.

The example programs illustrate some specific features: first, Basic09 programs can call and pass parame-

ters to and from programs written in assembly language or compiled by other compilers. The program Syscall (Code Example 1, page 34) is a 68000 program that allows a Basic09 program to perform system calls from within a procedure simply by passing a 68000 register image and an OS function code. Because it is reasonably closely tied to the calling syntax for OS-9 assembly-language system calls and outside the scope of this article, I have chosen not to discuss it here and simply treat it as a black box. Procedure ShowCall (Code Example 2, below) illustrates how Syscall would be invoked in a Basic09 procedure. In this case. I simply use the Syscall routine as a substitute for the high-level Basic09 OPEN command. The system call, which opens the file for read, passes back a path number in the d0 register, which is then assigned to the Basic09 variable Path.

The second example program, FormLetter (Code Example 3, page 36), illustrates a Basic09 procedure that adds a front end onto a standard text formatter. In order to personalize a form letter that I was preparing for a basketball team I coach, I wanted to incorporate names, addresses,

and greetings from a demographic file into the body of a standard letter. My local implementation of the formatter does not permit reading from outboard files into the standard data stream. I chose to put together tools that I already had at hand rather than do a "Swiss army knife" modification to the text formatter. (At this point, please realize that I am well aware of the case for making a permanent addition to the formatter but would have lost an example for this article.)

The technique I chose was to create a pair of named pipes, one of which receives a stream of data headed for the formatter, the other one receiving its output. I then connected those pipes to file paths within the Basic09 procedure via calls to the OS-9 shell. It was possible thereby to integrate my demographics with the body of my letter, sending the merged data as standard input to the formatter. In this example I could then send the output of the formatter to a file, but I could have just as easily invoked yet another pipeline to have it processed directly by my system's print spooler. (Basic09 commands to do so are enclosed as comments within the pro-

```
PROCEDURE ShowCall
 0000
             (* Procedure to demonstrate System Call from Basic09
 0001
            (* Define complex data type to represent registers TYPE registers=d0,d1,d2,d3,d4,d5,d6,d7,a0,a1,a2,a3:
 0036
 0069
                                                               INTEGER
 00A0
 00A1
             (* Declare necessary variables
             DIM Path: INTEGER
 00C0
             DIM Line:STRING[128]
 00C7
 00D3
             DIM IOPEN: INTEGER
 OODA
             DIM Regs:registers
 00E3
             DIM FileName: STRING[48]
 OOEF
             (* Initialize Variables
 00F0
 0108
             (* IOPEN is OS-9 System Call ISOPEN: Open path to a file
 0141
             IOPEN-$84
 0149
                This is the file we'll be reading from
 0173
             FileName="BOpen"
 017F
             (* Set up register image for system call
 0180
 01A9
             Regs.d0=1
             Regs.a0=ADDR (FileName)
 01B4
             RUN Syscall (IOPEN, Regs)
 01C2
 01D1
 01D2
             (* Assign returned path number to Basic09 variable
             Path=Regs.d0
 0205
 0210
             (* Read and print file contents WHILE NOT(EOF(#Path)) DO
 0211
               READ #Path, Line PRINT Line
 023C
 0246
             ENDWHILE
  024B
 024F
             (* Use Basic09 interface to close file
  0250
  0277
             CLOSE #Path
  027D
```

Code Example 2: Procedure to invoke Syscall

#### OS-9 OPERATING SYSTEM (continued from page 35)

gram.) These concepts can be extended in vastly more complex ways, all the while yielding the benefits of totally modular construction and interactive program development. The interactivity of Basic09 has left me consistently choosing it rather than C when portability is not a requisite, and I have found development to be much faster because of the interactive compilation, editing, and debugging.

#### Conclusions

I am tempted to call OS-9 the "poor man's Unix," even though I have to marvel at the power of its most sophisticated implementations, which are on par (or superior) to Unix running on a VAX in terms of speed and memory utilization. The concept of named, automatically located memory modules is an innovation that extends its capabilities into process control and real-time applications that are unsuitable for Unix, and its user interface and overall organization enable those familiar with Unix to adapt to its operation quickly. Its growing user base indicates that what was considered for many years an underground classic is now emerging from the shadows, and it should provide an interesting and productive environment for programmers of machines based on Motorola processors for years to come.

I would like to thank GMX Corp. for providing me with one of its Micro-20 systems for evaluating that implementation of OS-9.

#### Availability

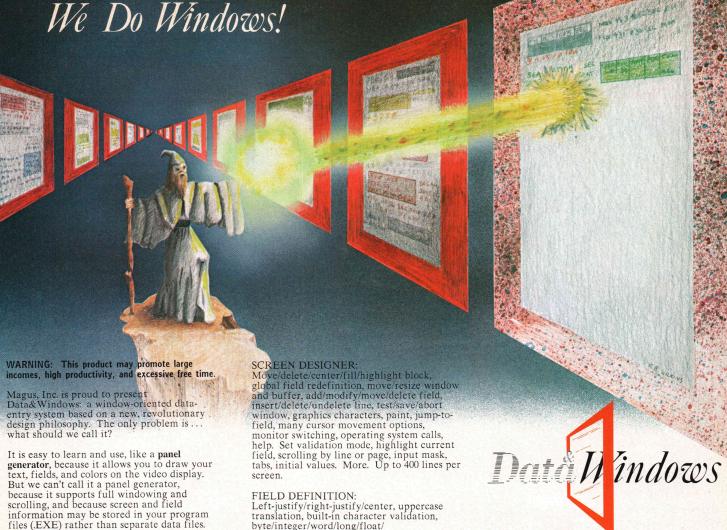
All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number and format (MS-DOS, Macintosh, Kaypro).

DDJ

Vote for your favorite feature/article. Circle Reader Service No. 4.

```
PROCEDURE FormLetter
0001
            (* Formletter: Add a front-end to K & P text formatter
            (* Written by Brian Capouch, 7/86
(* Define complex type to hold demographic info
0039
005C
008C
            TYPE NameRec=Name, Add1, Add2, Greeting: STRING[80]
00A8
00A9
            (* Declare variable storage
00C5
            DIM Record: NameRec
OOCE
            DIM NameFile, BodyFile: STRING[48]
OODE
            DIM Includer: STRING[10]
OOEA
            DIM NamPath, SplPipe, InPipe, OutPipe: INTEGER
OOFD
            DIM Counter: INTEGER
0104
            DIM Line:STRING[256]
0110
            DIM Index: INTEGER
0117
0118
            (* Trap out EOF coming down pipeline
013D
            ON ERROR GOTO 10
0143
0144
            (* Initialize variables
015C
            NameFile="Demographics"
            Includer=".us Body"
016F
017E
017F
            (* Set up files
0190
            OPEN #NamPath, NameFile: READ
019C
            GET #NamPath, Counter
01A6
01A7
            (* Set up output path to file
01C4
            CREATE #SplPipe, "LettersOut": WRITE
01D9
01DA
            (* Send each set to formatter, with preface and suffix
            FOR Index=1 TO Counter
021E
              (* Set up pipes to text formatter
(* This procedure "owns" this named pipe
0230
0252
027B
              CREATE #InPipe,"/pipe/in":WRITE
028E
              (* Now couple this pipe to text formatter
02B8
              SHELL "runb tformat </pipe/in >/pipe/out&"
02DF
              (* Open path to output pipe from formatter OPEN #OutPipe, "/pipe/out":READ
030A
              (* Next two lines commented out
031F
              (* They would send output to spooler instead of a file
(* SHELL "spl -nj </pipe/SplIn"
(* OPEN #SplPipe, "/pipe/SplIn": write
033E
0374
0394
03BA
              (* Now process data
03CE
              GET #NamPath, Record
03D9
              RUN PrintHeading (InPipe, "7/24/86")
03ED
              WRITE #InPipe, Record. Name
              WRITE #InPipe, Record. Add1
03FA
              WRITE #InPipe, Record. Add2
0407
0414
              WRITE #InPipe
041A
              WRITE #InPipe, Record. Greeting
0427
              WRITE #InPipe
042D
042E
              (* Send some commands to the formatter
0454
              WRITE #InPipe, ".fi"
0460
              WRITE #InPipe, Includer WRITE #InPipe, ".bp"
046A
0476
              CLOSE #InPipe
047C
047D
              (* Get output from formatter, send to spooler
04AB
              WHILE NOT (EOF (#OutPipe)) DO
04B6
                READ #OutPipe, Line
04C0
                WRITE #SplPipe, Line
04CA
              ENDWHILE
04CE
04CF 10
              (* We return here after each EOF trap
04F8
              CLOSE #OutPipe
04FE
           NEXT Index
           CLOSE #NamPath
CLOSE #SplPipe
0509
050F
0515
           END
0517
0518 100
           (* EOF on pipe generates OS-9 Error #211
0544
           ErrNo=ERR
054B
           IF ErrNo<>211 THEN
0558
              PRINT Beep;
              PRINT "System Error--->No. "+STR$(ErrNo)
055E
057B
             END
057D
           ELSE
0581
              GOTO 10
0585
           ENDIF
0587
           END
```

Code Example 3: Procedure to add front end to text formatter



It is flexible and powerful, like a libraryoriented programmer's toolkit, but you are not restricted to "visualizing" your data-entry windows as you type page after page of code to set up borders, fields, text and highlighting. Our innovative approach (called static windowing) eliminates the need for replication of static data in dynamic

It produces tight code, like a YACC (Yet Another Compiler Compiler), but you don't have to tolerate a myriad of small program modules that need to be compiled and maintained. Instead, our "screen designer" creates Microsoft object files which you simply link with your applications.

Add to this new, superior design philosophy the fact that it has more features, produces tighter code, and yields higher performance than any of the above. Throw in a clear, concise user manual, a thorough on-disk tutorial, and some example programs. Top it off with a utility program that documents each screen and another that allows you to prototype (or simulate) your application before you write a single line of code. Now, what would you call it?

Let's settle on a single word. Let's call it the "best."

But don't take our word for it. Order your demo disk today. You will receive a copy of the screen generator, the tutorial, and some documentation on the utility programs and library routines. Then make the decision yourself.

Or take advantage of our one-time introductory offer and get \$100 discount if you order before March 31, 1987.

Call (713) 665-4109 for more information. Major credit cards accepted. Left-justify/right-justify/center, uppercase translation, built-in character validation, byte/integer/word/long/float/ double/string/date field validation, retain data, auto-erase, protected fields, input required, use commas, use zeros/spaces, margin bell. User-defined character validations, pattern-matching validations, picture validations, and field types. More. Up to 9999 fields per screen.

LIBRARY ROUTINES:

Open, close, move, display, and refresh windows. Allow user to edit data fields in window, or to view and manipulate a window but not change data stored in it. Pull-down and pop-up menus. Read screen object file from disk. Intercept keyboard filter. Override default key actions. Automatic and manual refresh. Switch display device, erase all data fields on window, plot data onto fields or entire screens, retrieve data from fields or entire screens, sereen image dump, retrieve and modify screen and field attributes, locate field, force use of bios. Direct interfacing with some bios interrupts, including cursor and mouse control. More. Mnemonic and simple to use.

REQUIREMENTS:

IBM PC/XT/AT/JR or true compatible, DOS 2.0 or later, at least 128K free RAM, and the Microsoft C, Pascal, or Fortran compiler or the IBM C compiler. Support is available for other C Compilers and the XENIX operating system. Call for specifics.

IBM, IBM PC, IBM XT, and IBM AT are trademarks of International Business Machines. Microsoft and XENIX are trademarks of Microsoft Corporation.

Circle no. 336 on reader service card.

For More Information (713) 665-4109

C

Magus Inc.

Screen Designer

et's Do Windows!		
C Pasca Please send controductory discount four price	l □ FOI pies @ nt	\$345.00 -100.00 245.00
DataWindows wi ource Code ntroductory discount our price		\$695.00 - <b>100.00</b> <b>595.00</b>
Hurry! Introductory	offer expires	March 31, 1987
Show Me More! Send me a Demo		\$10.00
n Texas add 6.1259 Dutside U.S. add 15 Fotal enclosed	.00	
Enclosed is □ Check □ Visa Number Expiration Date		
Name Company Address City State		
Send to:		
Send to: MAGUS, INC. 4545 Bissonet Suit Bellaire, TX 7740	e #114	

## Chace

	253 main [variables] exter	
76	x[2][0]= .01;x[2][1]=.01;x[2][2	
77	x[3][0]=02;x[3][1]=.02;x[3][2	today:
78	<pre>printf("\n\nThe X matrix is");</pre>	today:
79	for(n1=0;n1 <a;n1++) td="" {<=""><td>x[0][</td></a;n1++)>	x[0][
80	for(n2=0;n2 <a;n2++)< td=""><td>y[0][i</td></a;n2++)<>	y[0][i
81	printf("\mxlxd.llxdl is xf	x1 ==
82	}	x3 < 1
83	/* slash is at left hand end */	n1 >
84	for(n1=0;n1 <a;n1++) th="" {<=""><th>n3 &gt;=</th></a;n1++)>	n3 >=
85	for(n2=0;n2(a;n2++) {	
86	if (n2==n1)	

unsigned char	1 0402	3
todays.month		9
todays.day		23
todays.year		86
x[0][0] changed value		
y[0][0] changed value		
x1 == 2.00000e + 00		
x3 < 8.30000e+00		
m1 > 9		
m3 >= 33		

#### MATRIX INVERSION

#### Run number is 1

The X matrix is x[0][0] is 1.000000 x[0][1] is 0.040000 x[0][2] is 0.030000 x[0][3] is 0.020000 x[1][0] is 0.020000

ptr	0x03fb
ptr->month	9
ptr->day	23
ptr->year	86
ptr->name[0]	<b>'</b> S'
ptr->name[1]	'e'
ptr->name[2]	'p'
ptr->name[3]	'\x00'
f	9.70865e-03
	9.99909e-01
x[0][0]	1.00000e+00

The perfect companion for MIX C has arrived. MIX C makes it easy to write C programs. Now Ctrace makes it easy to get them working. Introducing Ctrace, the exciting new C source debugger with animated trace.

#### **FUN AND EASY**

Ctrace makes it so easy to debug your C programs that you'll love doing it. You no longer have to mess with assembly language or hex addresses. Ctrace presents your program in a form that's instantly familiar. Your C source code is displayed just as you wrote it. All your variables are displayed just as you named them. And wait till you see your program in action. Ctrace brings it to life on the screen. You'll see your variable values changing as you watch your source code executing. Ctrace shows you how your program works, or why it doesn't work. After one session with Ctrace, you'll wonder how you ever programmed in C without it.

#### UNIQUE ANIMATED TRACE

Ctrace has a unique animated trace feature that shows you the flow of execution in vivid detail. Not just line by line, but statement by statement. It's like watching the bouncing ball as the cursor moves over your C source code, highlighting each statement as it executes. Press the space bar to execute one statement at a time, or press the return key and watch it go. It's exciting and educational. Who says learning has to be boring?

#### SIMPLE OPERATION

Ctrace is easy to operate too. Commands are executed with a single keystroke. Help screens are available if you forget a command. Pop up menus list command options. You simply position the cursor to the desired option and press the return key. Pop up messages alert you when anything important happens. To use Ctrace, simply compile your program with the trace option turned on. The executable program file is created as normal. Ctrace doesn't affect the size or the behavior of the program. You can execute your program with or without the help of Ctrace.

#### 4 VIEWS AT ONCE

Ctrace maintains 6 windows of information: source, output, variables, watch, symbols, and memory. You can view as many as 4 windows at the same time. The source window (top left) shows your C program. The output window (bottom left) shows the screen output from your program. The variable window (bottom right) shows all the variable names and values. The watch window (top right) shows the variables that you select along with any conditions vou've defined. The symbols window shows the addresses of variables and functions. The memory window shows any area of memory using data types that you select. Eight different screen layouts are available at the touch of a key. You can even define your own screen layouts.

#### COMPLETE PROGRAM CONTROL

Ctrace gives you complete control of your program. Execution options are single step, trace speed, and full speed. You can insert breakpoints on an unlimited number of statements. Execution is temporarily halted when a break point is hit. You can then

snoop around and see what your program has done to that point. You can even trace the flow of control backwards to see how your program got there. You can insert watch points on variable values. When the value of a variable satisfies the conditions you've defined, execution halts to let you examine your program. You can trace all functions or select just the ones you want to see.

#### THE RIGHT PRICE

If you could buy a debugger like Ctrace anywhere else you would expect to spend major bucks. Fortunately nobody else has a debugger like Ctrace. It's only available from MIX Software. And that's great news because you know our prices are right. Ctrace is an incredible value at only \$39.95. That's Right.



1132 Commerce Drive Richardson, Tx. 75081 (214) 783-6001



Source window with profile count showing number of times each statement has executed Pop up message indicates break point has been hit.



Source and variables windows shown side by side. Pop up message indicates that a watch point condition has been satisfied.



Source, variables, and memory window Memory window lets you view any area of memory using any data type



Change colors to suit yourself. Ctrace works with monochrome, color, Hercules, and EGA cards. Works on IBM compatibles and any computer with an IBM compatible BIOS

## *uourse*

#### THE C COMPILER

You can see that Ctrace is not your typical debugger. It's easy to understand and simple to operate. Likewise. MIX C is not your typical C compiler. It's small and fast. In fact it's the only full feature C compiler that can be operated comfortably on flopby disks. And as you would expect, MIX C is easy to use. It produces a complete program listing with all errors clearly identified and explained.

Although it's small, MIX C is not a subset. MIX C supports the full K&R standard, including the extensions that are often omitted in other C compilers. MIX C comes complete with a comprehensive 460 page book, a library of more than 175 functions, a blazingly fast linker, and tools for optimizing your programs for minimal space or maximum speed. All of this is yours for the incredibly low price of \$39.95. That's little more than the cost of most C books alone.

If you're just learning C, MIX C is your fastest, easiest, and cheapest way to master the language. If you've been frustrated by other C compilers. don't throw in the towel until you've tried ours. There's a world of difference. Our book includes a well written tutorial with lots of example programs. Our compiler includes the machine specific functions you need so you won't have to write them yourself. Compile and link operations take half as long with MIX C. That means you'll get your programs up and running twice as fast.

#### THE ASM UTILITY

Our ASM utility is available if you want to link assembly language functions to your C programs. It works with Microsoft's MASM or M80

assemblers. Macros make it easy! You can call assembly language functions just like C functions. You can even call C functions from assembly language. Lots of useful assembly language functions are included as examples. And the price is right at only \$10.

#### THE SPLIT-SCREEN EDITOR

Another great companion to the MIX C compiler is our split-screen editor. It makes writing programs even faster and easier. With the MIX Editor, you can compile, link, and execute your program at the touch of a key. Compiling is fast because the MIX C compiler reads the program directly from memory. Correcting errors is easy because the editor automatically positions the cursor to the first error in the program.

The MIX Editor works just like Micropro's WordStar. But through the magic of macros you can create your own custom version. You can map any key to any command. You can even define your own commands using the 100+ predefined commands. The split-screen feature is great for programming. You can edit two files at the same time and move text between files. It works great with any language. It has automatic line numbering for BASIC. It has auto indent for structured languages like Pascal and C. It even has fill and justify for English. All these features and more are yours for the incredibly low price of \$29.95.

#### THE MIX C WORKS

The combination of Ctrace with MIX C makes C programming a real joy. MIX C provides the power of a compiler while Ctrace provides an execution environment that's more

elegant than an interpreter. Add the ASM utility and our versatile splitscreen editor to the package and you've got a terrific C programming system. We call it the MIX C Works. What's great is that you can buy all four products for a fraction of the cost of other C compilers alone. Yes, buy all four and we'll give you a big \$29.95 discount off our already rock bottom prices. Only \$89.90 for the MIX C Works. Now that's a deal. That's Right.

#### TO ORDER CALL TOLL FREE: 1-800-523-9520

For technical support and for orders inside Texas please call (214) 783-6001

Or Contact one of our Worldwide

Distributors direct in:

Canada: Saraguay France: Info/Tech Australia: Techflow Switzerland: DMB

1-800-387-1288 1-43-44-06-48 047-586924 CH-523-31817

System Requirements
Editor, C Compiler, & ASM Utility
MSDOS/PCDOS 2.0 or higher 128K Memory

1 Disk Drive

or CP/M 2.2 or higher (Z80)

1 Disk Drive (2 recommended)

System Requirements

MSDOS/PCDOS 2.0 or higher IBM compatible BIOS 256K Memory 1 Disk Drive

## MIX C WORKS Only

Price Product . (\$39.95) \$ \_\_\_ Ctrace \_C Compiler . . . . (\$39.95) \$ \_ASM Utility . . . . . (\$10.00) \$ \_ \_Split-Screen Editor . (\$29.95) \$ \_\_ \_\_The MIX C Works (\$89.90) \$ \_\_\_\_ (includes all of above) Texas Residents Add 6.125% Sales Tax ..... Add Shipping Charges . . . \$ . In USA: add \$5 per order In Canada: add \$10 per order Overseas: add \$10 for editor add \$20 for compiler add \$30 for Works

Total of Your Order .... \$

#### 30 Day Money Back Guarantee **Not Copy Protected**

Please check method of payment \_\_Check \_\_Money Order \_\_MC/VISA Card # Expiration Date

Please check operating system

Please give name of computer

MSDOS/PCDOS \_\_ CP/M

Please check disk size

\_ 51/4" \_ 31/2" \_ 8"

Please check disk format if CP/M

\_SSSD \_SSDD \_DSDD

Your Name \_\_

Street \_\_\_

State Zip \_\_\_

Telephone (\_\_\_\_\_) \_\_\_\_-\_

Country \_\_\_

1132 Commerce Drive Richardson, Tx. 75081 (214) 783-6001

Ask about our volume discounts!

Dealer Inquiries Welcome

## Macintosh Buttons and Amiga Gadgets

by Jan L. Harrington

hen you put aside emotional reactions to the Macintosh and Amiga computers to take a more objective look at the two machines, it appears that, at least in concept, they are more alike than different. In addition to similarities in user interface standard, both use the 68000 microprocessor. Both have operating systems that are partially in ROM and partially on disk. Programmers who wish to adhere to the standard user interface make use of a set of system routines to create and manage windows, menus, graphics, and text.

The functions that the Macintosh and Amiga system routines provide are not equivalent, however. The Macintosh, for example, provides text handling routines not found on the Amiga. By the same token, the Amiga libraries contain animation routines not found on the Macintosh. The disparity between the functions provided by system routines presents challenges for programmers, especially if they are attempting to adapt software from one machine to the other.

This article explores the details of the standard Macintosh and Amiga user interfaces, examines the system routines programmers use to create those interfaces, and discusses those system routines through a pair of sample 68000 assembly-language programs that support a portion of the standard user interfaces.

Jan L. Harrington, 4002 Stearns Hill Rd., Waltham, MA 02154. Jan is an assistant professor in the Computer Science Department at Bentley College. She is the author of Macintosh Assembly Language: An Introduction. The Mac's system routines support its user interface more completely.

#### **User Interface Standards**

The Macintosh standard user interface has caused many people to redefine what they mean when they say software is easy to use. You expect to be able to run a Macintosh program by double-clicking its icon on the Macintosh desktop. You expect to find at least Apple, File, and Edit menus present, and you expect those menus to behave in a consistent manner (they should "pull down" to expose the menu items, some of which may be associated with a keyboard equivalent). You expect to be able to use scroll bars to move throughout a document and to be able to simulate a click on an OK button by pressing the Return key. The Macintosh user interface standard is clearly defined in Chapter 2 of Inside Macintosh.1 the extensive technical documentation for the machine.

The Amiga also supports a mouse-driven interface. Amiga applications that have icons can be run by double-clicking that icon from the Workbench window with the left mouse button. Amiga menus also pull down, the result of right mouse button action. Menu items can be associated with keyboard equivalents, and Amiga software has appeared with scroll bars.

The Amiga interface standard, however, is not as strictly defined as is the Macintosh standard. Suggestions for the interface do appear in the *Intuition*<sup>2</sup> manual, the part of the technical documentation that describes the routines supporting the windowing environment.

The window is central to both the Macintosh and Amiga. You can move windows about the screen (that is, within the same plane), move them relative to other windows on the screen (that is, from front to back), size them, and close them. All of these functions are initiated by mouse action on some portion of the window. Window movement within the plane is controlled by the drag region, that part of the window's title bar not taken up by the window title or other graphics. A click of the mouse button on a box at the far left of the title bar will close the window (the Macintosh calls it a "GoAway box," the Amiga a "CloseWindow gadget"). The Amiga window title bar also contains, at the far right, gadgets that move the window from front to back ("depth arrangement gadgets"); Macintosh windows move back one layer when another window is activated and brought to the front. In both machines, windows are sized with the overlapping boxes that appear in the lower-right corner of the window (the Macintosh calls it a 'grow icon;" the Amiga calls it a "sizing gadget").

Both the Macintosh and Amiga collect information and give warnings using special windows. The Macintosh uses "dialog boxes" to collect information and "alerts" to give warnings. For example, a dialog box will appear to accept the name under which a file should be saved, and an alert box appears if you attempt to close a document window without

saving its contents. The Amiga uses "requesters" to collect information and "alerts" to let you know that something catastrophic has occurred. Requesters are analogous to dialog boxes, but whereas Macintosh alerts signal that a potential for harm occurs, Amiga alerts appear only after it's too late to recover. Amiga alerts are generally equivalent to Macintosh system alerts (generated by 68000 system errors). Usually you close requesters, dialog boxes, and Macintosh alerts by clicking on a small rectangle containing a message such as OK or Cancel. (The Amiga calls the small rectangles "gadgets"; the Macintosh calls them "buttons," which are a type of "control.")

Menus, too, are essential to both the Amiga and Macintosh user interfaces. Macintosh menu titles are visible at all times across the top of the screen in the "menu bar." The Amiga "menu strip," which also appears across the top of the screen, is visible only when you press the right mouse button.

Menus on both computers pull down; the menu choices appear in a box below the menu title as you drag the mouse pointer over them. Amiga menu items may also have submenus, which appear when the mouse pointer is dragged over the menu item. In either machine, menu items can be associated with keyboard equivalents—a single character that, when pressed in conjunction with a special modifier key (the command key on the Mac, the solid A key on the Amiga), simulates the selection of a menu item with the mouse.

Both computers maintain menu definitions as linked lists of data structures containing data needed to draw the menus. At any one time, the Macintosh supports only one menu list. You can make changes by inserting and removing menus from the list. The Amiga logically attaches menu lists to windows rather than to the screen, making it possible for many menu lists to be present in memory at any given time. Which menus appear when you depress the right mouse button therefore depends on which window is active at the time.

A Macintosh program that adheres to the standard user interface has at least three menus. The Apple menu appears at the far left of the menu bar; its title appears as an apple icon (an apple with a bite taken out of it). The Apple menu supports the Macintosh desk accessories. The second menu from the left is the File menu. The File menu opens, closes, saves, and prints files and exits from the program to the operating system. The third menu, the Edit menu, handles the text editing functions—Cut, Copy, Paste, Clear, and sometimes the Undo function as well.

The Amiga does not provide routines to implement its own standard interface recommendations.

Amiga programs that follow the interface suggestions made in the *Intuition* manual provide at least two menus in each menu strip. The leftmost menu is the Project menu, which is analogous to the Macintosh File menu—it manages opening, closing, saving, and printing of files as well as exiting from the program to the operating system. The second menu from the left is an Edit menu, which provides access to the same editing functions as the Mac's Edit menu.

Text editing standards are also an important part of both the Macintosh and Amiga user interfaces. Macintosh programs that support text entry of any kind, including even the entry of a single line into a dialog box, support cut, copy, and paste operations from a standard Edit menu. The editing functions are also present in programs that do no text editing because desk accessories use cut, copy, and paste even if an application does not. Amiga programs that require text entry (most notably word processors) support the same text editing functions as Macintosh programs do, but the absence of desk accessories means that Edit menus do not appear in programs that are not concerned with text.

#### Implementation

To explore the support the Macintosh and Amiga provide for their standard user interfaces. I wrote two programs in 68000 assembly language, one for each machine (see Listings One and Two, pages 64 and 69 for the Macintosh program and Listing Three, page 69, for the Amiga program). The programs are more or less equivalent in function. Each opens a window for text entry (the Amiga program also opens a custom screen) and creates menus (three for the Macintosh, two for the Amiga). Each supports the entry of text from the keyboard. The programs both terminate if you either select Quit from the appropriate menu or click the mouse pointer on the box that closes the window. Both also make extensive use of constants and data structure offsets from the include files supplied with the Macintosh 68000 Development System and the Macro Assembler, Amiga respectively.

The Macintosh system routines are invoked through the 68000's trap mechanism (all system calls are assembled to begin with %1010, which is then trapped by the microprocessor). The trap mechanism retrieves the actual location of the routine from a jump table, which is loaded from ROM to RAM at system start-up. Calls to system routines are performed with trap macros, all of which begin with an underbar (for example, \_GetRMenu). The trap macros themselves are defined in the Macintosh's include files. Macintosh system routines are organized into "managers," which must be initialized before the routines are called. Lines 5-14 of Listing One initialize all the Macintosh managers.

Amiga system routines are contained in libraries. All libraries except the exec must be opened before they can be used. This includes the intuition library, which is opened at the top of Listing Three with a call to the system routine *OpenLibrary* (lines 44–51). Each library has a base address. Exec's base is fixed and assigned to the constant \_AbsExecBase; all other library bases are returned by *OpenLibrary*. System routines are

MAC BUTTONS, AMIGA GADGETS (continued from page 41)

called as subroutines whose starting addresses are relative to the library base, which is placed in register A6 before the call. Assuming that the correct address is in A6, calls to Amiga system routines are handled by the macro callsys (lines 4—6).

The functional differences between the two programs are the result of the system routines available on the two computers. The Macintosh program, for example, has a menu the Amiga program does not. The Apple menu is provided for the Macintosh's desk accessories. Implementation of the desk accessories is handled completely by a series of calls to system routines. The text in the Macintosh window will word wrap because the routines that manage the text editing environment automatically handle that function. The editing operations—cut, copy, and paste-have also been implemented because each is handled by a call to a single system routine.

The Amiga does not support desk accessories and so has no equivalent to the Macintosh's Apple menu. The Amiga also does not provide system routines for doing word wrap or the standard editing functions. Therefore, text entry in the Amiga window is exactly like using a typewriter. On the other hand, the Amiga window can be sized, moved about the screen, and moved back and forth in the plane. These functions are handled automatically by the Amiga's operating system; they do not need to be included in an application's code. Performing the same operations on the Macintosh requires including additional program code (calls to system routines).

The Amiga program is more than twice as long as the Macintosh program. There are two major reasons for this. First, the Macintosh can obtain parameters for creating data structures (for example, windows and menus) from a resource file. A resource file contains templates that describe the contents (for example, items to appear in a menu), location (for example, initial coordinates of a window), and characteristics (for example, menu items that should initially be disabled) of structures an ap-

plication will use. The Macintosh program's resource file (Listing Two) contains templates for the three menus and one window. Second, although both the Macintosh and the Amiga maintain data about program objects in linked lists of data structures, Macintosh system routines perform most of the structure initialization and list management, whereas the Amiga leaves those functions to the programmer. The amount and type of programming required to achieve the same program function is therefore rather different on the two machines. You can see examples of these differences, especially in terms of creating menus and windows, when the two sample programs perform event trapping and do text I/O.

#### **Creating Menus**

The Macintosh can create a menu by calling the system routine GetRMenu. GetRMenu reads the template from the resource file, allocates space in RAM for the menu record, and loads the data into the appropriate locations in that menu record. Menu items are stored in a linked list. The routine requires one parameter (the resource file ID that identifies the menu) and space on the stack for a handle to the menu record. After the routine is called, an application pulls the handle from the top of the stack, where it has been placed by GetR-Menu. Initializing a menu record therefore requires four lines of code. The template for the Macintosh program's File menu, for example, appears in lines 5-14 of Listing Two. The menu structures are actually initialized in lines 29-32 of Listing One.

Initializing a menu data structure for an Amiga program is considerably more complex. Assuming that the menu items are to be text rather than graphics, the text must first be loaded into intuition text structures. Each intuition text structure must be included in a menu item structure. The menu item structures are then assembled into a linked list whose head is incorporated into the menu data structure.

In the program in Listing Three, the intuition text structures are initialized in the subroutine *SetText* (lines 257–265), which is invoked by the macro *passtext* (lines 10–14). For

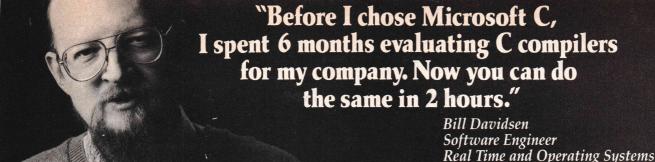
each menu item, the text must be included in the program as a constant. The program must also allocate space for the intuition text structure (see lines 354—393 for the data structures associated with the Amiga program's menus). The intuition text structures for the Project menu, for example, are handled in lines 92—98.

Once the intuition text structures have been initialized, the menu items are initialized and chained into a linked list. The actual initialization is performed by the subroutine *SetItem* (lines 266–277), which is invoked by the macro *passitem* (lines 15–22). It is up to the programmer to allocate storage for each menu item data structure and to load the list pointers correctly. The Project menu's menu items are initialized and linked in lines 99–111.

The final step in the process is the initialization of the menu data structure. All the menus that will be present in a single menu strip are maintained in a linked list. Therefore, the initialization must include setting a pointer to the next menu in the list. The programmer must also determine coordinates for the physical position of the menu in the menu strip. The Amiga program sets up the Project menu in lines 112—123. After the Project menu has been completed, the entire process is repeated for the Edit menu.

The Macintosh requires the use of a separate system routine to insert menus into the linked list of menus that will appear in the menu bar at any given time. *InsertMenu* handles the insertion of a menu into the menu list. A programmer supplies either the handle of a menu after which this menu should be inserted or a parameter indicating that this menu should be last (that is, rightmost). Lines 33—35 of Listing One, for example, insert the File menu into the Macintosh's menu list.

Merely inserting a menu into the menu list does not, on either machine, display the menu. The Macintosh routine *DrawMenuBar* (see line 43 in Listing One) is equivalent in function to the Amiga routine *SetMenuStrip* (lines 149—152 in Listing Three). The Macintosh routine actually displays the menus currently in the menu list on the menu bar; the Amiga routine attaches the menu list to a window so



"Call us. You can get Microsoft C or our comprehensive report on C by the day after tomorrow."

> Bruce Lynch, President The Programmer's Shop

> > THE C TEST

The security of thorough research. It took Bill Davidsen six months to thoroughly evaluate all C products before he selected Microsoft C. For him, its tight code and UNIX System V<sup>™</sup> compatibility were exactly what he needed. And now Version 4.00 includes CodeView,™ a source-level windowing debugger.

Thanks to expert users like Bill, and The Programmer's Shop, you can enjoy that satisfied feeling of thorough

product evaluation in just a few hours.

We recommend evaluating software by also getting detailed information from several different sources, including unbiased reports and reviews. Bill agrees completely.

In fact, he helped us compile the objective opinions of 4 magazines, 14 users and 3 industry analysts in a 16-page report on C: The C Test. It can help you be absolutely sure of making the choice that's best for you. And it's absolutely free.

C for yourself. As an objective evaluation by users and professionals alike, The C Test is one of the most comprehensive and informative reports currently available on

C development tools. It's only available from The Programmer's Shop. And it's yours free for the asking. Here's what you'll find in it:

The C Test • Detailed Tech Specs Benchmark Source Code
 Magazine Reviews • Users' Feedback • Performance Benchmarks • User Study and Profiles • Test Drive Survey Results

37 Compatible Products

And if you're looking for even more C support, Microsoft-compatible libraries for file management, graphics, screen control, object-oriented programming and other tools are ready to ship.

The best programs for less. We think the only way to serve you is to give you the best programming alternatives. The best recommendations for your needs. To deliver immediately. And this is how we do it.

We start by giving you a choice of over 62 programming language implementations and 174 support programs. All from the same source. All competitively priced.

Our informed programmers offer free advice whenever

you call with any questions about any product.

And when you place an order, we can rush it to you in 48 hours or less. That's the kind of service and support our 10,000 customers have come to expect.

Because we've become a success by giving the best

advice for free and selling the best software for less.

To order Microsoft C(\$279) or for your free copy of *The C Test,* simply call the toll-free number below:

**1-800-421-8006.** In Massachusetts, call 1-800-442-8070.

#### MICROSOFT. C Compiler Version 4.00

MICROSOFT C COMPILER

 Produces fast executables and optimized code including elimination of common sub-expressions. NEW!

Implements register variables

Small, Medium and Large Memory model libraries

- Compact and HUGE memory model libraries. NEW!
   Can mix models with NEAR, FAR and the new HUGE pointers.
   Library routines implement most of UNIX System V C library.
- Start-up source code to help create ROMable code. NEW!
- Full proposed ANSI C library support (except clock). NEW!
   Link your C routines with Microsoft FORTRAN (version 3.3 or higher), Microsoft Pascal (version 3.3 or higher) or Microsoft Macro Assembler.
- Microsoft Windows support and MS-DOS 3.1 networking support.

#### MICROSOFT PROGRAM MAINTENANCE UTILITY. NEW!

- Rebuilds your applications after your source files have changed.
- Supports macro definitions and inference rules.

#### OTHER UTILITIES.

- Library Manager.
- Overlay Linker.
- EXE File Compression Utility.
- EXE File Header Utility.

#### MICROSOFT CodeView

WINDOW-ORIENTED SOURCE-LEVEL DEBUGGER. NEW!

- Watch the values of your local and global variables and expressions as you debug.
- Set conditional breakpoints on variables, expressions or memory; trace and single step.
- Watch CPU registers and flags as you execute.
- Debug using your original source code, the resulting disassembly or both intermingled.

Microsoft C comes with a 30-day money-back guarantee from The Programmer's Shop.

UNIX System V is a trademark of AT&T Bell Laboratories.

Microsoft is a registered trademark and CodeView is a trademark of Microsoft Corporation.

#### THE PROGRAMM!

The programmer's complete source for software, services and answers.

(617) 826-7531 128 Rockland Street, Hanover, MA 02339

#### MAC BUTTONS, AMIGA GADGETS (continued from page 42)

that, when the window is active and the right mouse button is depressed, the menu strip appears.

There are both advantages and drawbacks to the Amiga's way of handling menus. The major drawback is the burden the Amiga places on the programmer. Establishing Amiga menus requires a great deal of work to initialize data structures and a great deal of care to ensure that pointers are stored properly. The Macintosh, on the other hand, isolates the programmer from dealing directly with the data structures and from list management. The trade-off is flexibility. Macintosh menu items are restricted to text (though a limited number of icons can appear with them), whereas Amiga menu items can be graphics (the intuition text structures can be replaced with graphics data structures). A Macintosh programmer has control only over the relative placement of a menu in the menu bar; an Amiga programmer can determine exactly where a menu should appear.

#### **Creating Windows**

Macintosh windows can also be defined by templates in resource files (see lines 24–29 in Listing Two). An application can then create the window by pushing space for a window pointer on the stack, pushing three parameters, and calling *GetNewWindow* (see lines 44–49 in Listing One). The programmer must provide storage for the window record and for its pointer (lines 195–196).

Because the Amiga doesn't support resource files, an Amiga program must load a window data structure explicitly before calling the *Open-Window* system routine, which actually creates the window (lines 69–91 in Listing Three). If the window is to appear in a custom screen rather than the default Workbench screen, the program must also first initialize a screen data structure and call *Open-Screen* (lines 52–68). Note that al-

though it doesn't matter to the Macintosh whether windows or menus are created first, it does matter to the Amiga. Amiga menus are attached to windows, not to the screen, and therefore a menu strip is useless unless a window has previously been created to which it can be attached.

#### **Event Trapping**

Event trapping on the Macintosh and Amiga is similar in principle but somewhat different in detail. The general idea is to somehow let the computer know which events are of importance and then to enter a wait state until a desired event occurs. Once an event has been recorded, a program must identify which type of event has been posted and take action based on that particular event.

The Macintosh posts events to a system event queue. Events of interest to an application program (that is, those that aren't handled automatically by the system) are passed on to the program. An application program checks its queue repeatedly

#### **METACOMCO**

The quality source for Atari ST & Amiga software





#### THE SYMBOLIC LANGUAGE FOR ATARI ST and AMIGA

An interpreter/compiler providing a complete LISP development environment for \$199.95.

-also available-

Macro Assembler - Professional development system
Amiga - \$ 99.95
BCPL · NEW! Full standard BCPL compiler · ST\$149.95
Lattice 'C' - The well known Lattice 'C' compiler - ST & Amiga \$149.95
Cambridge Lisp - The interpreter/compiler - ST & Amiga \$100.05
MCC Pascal - Fast ISO/ANSI standard compiler - ST & Amiga \$ 00.05
Metacomco MAKE - NEW! UNIX-like MAKE utility for the ST \$ 60.05
MENU + - Best selling SI MENU generator \$ 20.05
Metacomco SHELL - NEW! Amiga's intelligent programming shell \$ 60.05
Metacomco TOOLKIT - Smartest tools available for the Amiga\$ 49.95
Cambridge LISP - CP/M-68K - \$295. Call for Sinclair QL products. Languages come
with full documentation, libraries & screen editor. ST languages include MENU+ and pro-
vide full interfaces to GEM VDI and AES functions. Metacomco provides experienced

technical support and keeps its customers informed of new products and upgrade releases.

METACOMCO

5353 #E Scotts Valley Dr. Scotts Valley, CA 95066 Contact your local dealer or call: Tel. (US) 800-AKA-META (CA) 800-GET-META BIX: mhill Compuserve: 73247,522 Add 6½% tax if CA resident

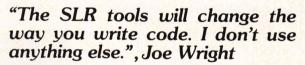
Registered trademarks: Lattice - Lattice, Inc; Atari ST Atari; UNIX - Bell Labs; Amiga - Commodore Amiga





Circle no. 358 on reader service card.

Does this look familiar?
What if each change
you made to your
program was ready to
test in seconds instead
of minutes?



RELOCATING MACRO ASSEMBLERS • Z80 • 8085 • HD64180

- Generates COM, Intel HEX, Microsoft REL, or SLR REL
- Intel macro facility
- All M80 pseudo ops
- Multiple assemblies via command line or indirect command file
- Alternate user number search
- ZCPR3 and CP/M Plus error flag support, CP/M 2.2 submit abort
- Over 30 user configurable options
- Descriptive error messages
- XREF and Symbol tables
- 16 significant characters on labels (even externals)
- Time and Date in listing
- Nested conditionals and INCLUDE files

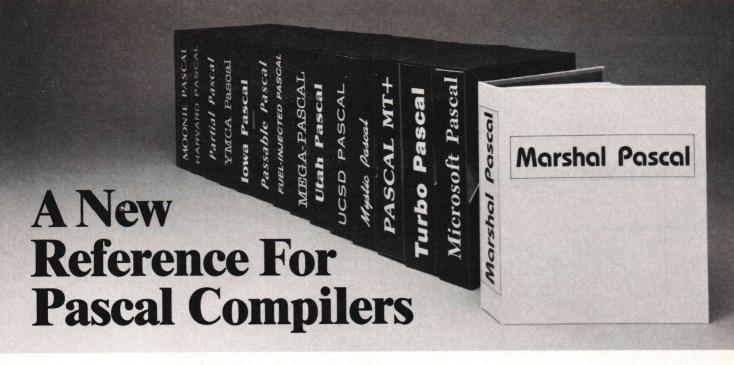
Supports math on externals requires Z80 CP/M compatible systems with at least 32K TPA

SLR Systems

1622 N. Main St., Butler, PA 16001

(412) 282-0864 (800) 833-3061

Circle no. 78 on reader service card.



	Acke	rman	Sie	eve	I	/O	Gauss	s-Seidel	Floatin	g Point
Marshal Pascal	11.9	5.1K	4.8	3.4K	1.9	6.8K	5.7	11.5K	10.5	5.3K
IBM Pascal	12.4	34.7K	11.7	27K	2.6	28K	7.6	34.6K	10.5	21K
Turbo Pascal	22.7	11.6K	14.2	11.5K	2.2	12.5K	4.7	13.5K	28	11.4K
Microsoft C 4.0	15.9	9.3K	5.8	6.5K	1.9	8.9K	6.0	23.6K	33	19.6K
	Sec.	Code Size	Sec.	Code Size	Min.	Code Size	Sec.	Code Size	Sec.	Code Size

#### **Unparalleled Speed and Power**

Marshal Pascal™ is the most highly code-optimized Pascal compiler for PCs. Period. In fact, Marshal Pascal produces code so fast and compact that even the most efficient C compilers fall behind in performing many operations. Also, Marshall Pascal is an ISO implementation thus offering portability to other computer environments. You may address as much memory as your operating system allows and a variety of memory models are supported. Among the useful extensions included are: separate compilation of modules (both Modula-2 and Pascal forms), structured constants and structured function values, variable-length string types and procedural parameters.

#### Turbo Pascal® Translator

Our translator brings your Turbo Pascal software over to an ISO/Marshal-readable format. You can watch your present Turbo programs run in a fraction of their former time and code space!

#### 8087-80287 Support

Marshall Pascal supports the Intel 8087-80287® math processors *inline*. If you don't have the math chip, then '87-287 code emulation is a provided option.

#### Marshal Language Systems

1136 Saranap Ave., Ste. P, POB 2010, Walnut Creek, CA 94595

#### **Microsoft Linkability**

Marshal Pascal's true relocatable linker allows you access to the Microsoft family of languages and assemblers. A flexible object code librarian and powerful overlay capabilities are also included.

#### **Powerful Compile Options**

Marshal Pascal gives you a number of compile options, including an optimization by-pass for speedier compiles, I/O "fine-tuning", constant folds and a syntax evaluator just to name a few. A wealth of compile-time checks permits you to find the more subtle logic errors, reducing debugging time enormously.

Space does not permit us to list all the features of Marshal Pascal, all of which are provided USER options, not additional COST options. Marshal Pascal is not broken apart to make many products out of what should be only one good one.

The Price? \$189. includes everything.

Supports PC-DOS®, MS-DOS®, CP/M-86® and Concurrent DOS®.

To order your copy of **Marshal Pascal**, call: (800) 826-2222 In California: (415) 947-1000

Marshal Pascal is a TM of Marshal Language Systems. Pascal MT + . CP/M-86 and Concurrent DOS are ⊗ of Digital Research, Inc.
IBM Pascal, PC-DOS are ⊗ of IBM Corp. Turbo Pascal is a ⊗ of Borland International, Inc. Microsoft TM, Microsoft C and MS-DOS
are ⊗ of Microsoft Corp. Mystic Pascal is a ⊚ of Mystic Canynon Software. Intel 8087/80287 is a TM of Intel Corp. UCSD Pascal is a
TM of Peccar Software Systems. Utah Pascal is a TM of Ellis Computing, Inc.

Circle no. 317 on reader service card.

## SAS Institute Inc. Announces

## Lattice C Compilers for Your IBM Mainframe

Two years ago...

SAS Institute launched an effort to develop a subset of the SAS® Software System for the IBM Personal Computer. After careful study, we agreed that C was the programming language of choice. And that the Lattice® C compiler offered the quality, speed, and efficiency we needed.

One year ago...

Development had progressed so well that we expanded our efforts to include the entire SAS System on a PC, written in C. And to insure that the language, syntax, and commands would be identical across all operating systems, we decided that all future versions of the SAS System—regardless of hardware—would be derived from the same source code written in C. That meant that we needed a C compiler for IBM 370 mainframes. And it had to be good, since all our software products would depend on it.

So we approached Lattice, Inc. and asked if we could implement a version of the Lattice C compiler for IBM mainframes. With Lattice, Inc.'s agreement, development began and progressed rapidly.

Today...

Our efforts are complete—we have a firstrate IBM 370 C compiler. And we are pleased to offer this development tool to you. Now you can write in a single language that is source code compatible with your IBM mainframe and your IBM PC. We have faithfully implemented not only the language, but also the supporting library and environment.

Features of the Lattice C compiler for the 370 include:

- Generation of reentrant object code. Reentrancy allows many users to share the same code. Reentrancy is not an easy feature to achieve on the 370, especially if you use non-constant external variables, but we did it.
- Optimization of the generated code. We know the 370 instruction set and the various 370 operating environments. We have over 100 staff years of assembler language systems experience on our development team.
- Generated code executable in both 24-bit and 31-bit addressing modes. You can run compiled programs above the 16 megabyte line in MVS/XA.
- Generated code identical for OS and CMS operating systems. You can move modules between MVS and CMS without even recompiling.
- Complete libraries. We have implemented all the library routines described by Kernighan and Ritchie (the informal C standard), and all the library

routines supported by Lattice (except operating system dependent routines), plus extensions for dealing with 370 operating environments directly. Especially significant is our byte-addressable Unix\*-style I/O access method.

■ Built-in functions. Many of the traditional string handling functions are available as built-in functions, generating in-line machine code rather than function calls. Your call to move a string can result in just one MVC instruction rather than a function call and a loop.

In addition to mainframe software development, you can also use our new cross-compiler to develop PC software on your IBM mainframe. With our cross-compiler, you can compile Lattice C programs on your mainframe and generate object code ready to download to your PC.

With the cross-compiler, we also offer PLINK86™ and PLIB86™ by Phoenix Software Associates Ltd. The Phoenix linkeditor and library management facility can bind several compiled programs on the mainframe and download immediately executable modules to your PC.

#### Tomorrow...

We believe that the C language offers the SAS System the path to true portability and maintainability. And we believe that other companies will make similar strategic decisions about C. Already, C is taught in most college computer science curriculums, and is replacing older languages in many. And almost every computer introduced to the market now has a C compiler.

#### C, the language of choice...

C supports structured programming with superior control features for conditionals, iteration, and case selection. C is good for data structures, with its elegant implementation of structures and pointers. C is conducive to portable coding. It is simple to adjust for the size differences of data elements on different machines.

#### Continuous support...

At SAS Institute, we support all our products. You license them annually; we support them continuously. You get updates at no additional charge. We have a continuing commitment to make our compiler better and better. We have the ultimate incentive—all our software products depend on it.

#### For more information...

Complete and mail the coupon today. Because we've got the development tool for your tomorrow.

SAS.

SAS Institute Inc. SAS Circle, Box 8000 Cary, NC 27511-8000 Telephone (919) 467-8000 x 7000

<ul> <li>□ the C compiler for MVS softy</li> <li>□ the C compiler for CMS softy</li> <li>□ the cross-compiler with PLIN</li> </ul>	ware developers	
todayso l'il be read		w.
Please complete or attach your b	이 구설에 가셨는 생산하다면 하면 회사하다	
Name		
Title		
Company		
Address		
City	State	ZIP
Telephone		

#### MAC BUTTONS, AMIGA GADGETS (continued from page 44)

with the routine GetNextEvent (lines 62-65 in Listing One) to determine if an event has been posted. If GetNext-Event returns a Boolean result of FALSE, then the program simply branches to the top of the event loop (line 59) to check again. Assuming that an event has been posted, information about the event is stored in an event record. The program can then use information from that record to identify the exact type of event (see lines 69-74). In some cases, events not of interest to the application can appear in the event queue. When that occurs, the program simply ignores the event (line 75). When a desired event is identified, however, Macintosh programs generally branch to submodules, each of which processes a single event type. When the event has been handled, the program returns to the event loop to idle until another event is posted to the event queue.

The Amiga reports events via message ports, which must be initialized before they can be used. The *Open-Window* routine creates an intuition message port, but ports must be created explicitly for the console device, which will be used for text I/O. The subroutine *CreatePort* (lines 278–307 in Listing Three) allocates a signal bit for a new port, allocates memory for the port's data structure, initializes a task control block for the port, and adds the port to the linked list of current message ports.

Amiga programs do not need a program loop to idle while waiting for an event. Instead, they can use the system routine *Wait*, which idles until a desired event occurs. *Wait* must be supplied with the signal bits assigned to each of the input ports that should be monitored for events. In the sample Amiga program, that includes the signal bit for the intuition message port and the signal bit for the console read port (see lines 184–193 in Listing Three).

Unlike the Macintosh, the Amiga

doesn't report all types of events automatically. In line 77, the system is instructed to report only two of the types of events that may occur when this window is active—a click in the window close gadget and a selection from a menu. Therefore, any event detected by *Wait* should be an event useful to the program.

If a Macintosh program identifies a selection from a menu (a "mousedown" event in the menu bar), the program is faced with the problem of identifying which item from which menu has been selected. A single system routine, MenuSelect (lines 125-127 in Listing One), returns both the menu's resource ID and the number of the menu item. MenuSelect uses a field from the event record as input—the coordinates of the mouse pointer when the selection was made. The menu number, returned in the high-order word of the longword result, is then isolated from the menu item, which is returned in the low-order word (lines 128-130). Finally, the menu number can be used

## C Bricklin Run

#### C-scape: A Prototyping Tool

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy of **Dan Bricklin's Demo Program.** Period. No exceptions." Soft-letter

**C-scape** should immediately be included in that quote. Use Dan Bricklin's Demo Program to design exactly how each screen should appear. You can even use Demo to capture existing screens, making language conversion projects easier. Then use **C-scape's demo2c** utility to turn each screen into C code, complete with text, menus, input fields, masking, and colors. Screens will compile the first time. Then link in **C-scape's library** functions for scrolling, validation, type support, and much, much more, and presto—your program is done.

#### C-scape: A Screen Generator and Library

"Dan Bricklin's Demo Program without C-scape is like VisiCalc without a printing function." Joseph Katz, C Notes, Sextant Magazine

C-scape is by far the most flexible and easiest to learn and use screen library available. It offers over 100 useful functions, and its unique design makes it easy to create your own functions with maximum support and minimum constraint. It's a veritable function construction kit. Combined with Dan Bricklin's Demo Program, C-scape will accelerate your programming productivity, boost your creativity, reduce coding errors, and facilitate maintenance. No more screens from scratch. No more procrustean screen libraries.

Fully definable validation • full type support • automatic scrolling • full printf % substitution inside field definitions • Lotus-like/pop-up/pull-down menus • menus, data entry fields, windows, prompts, text • supports standard C types • full color support • completely definable keys • full screen I/O support • user-definable field types • source code included • input masking • field marking • orthogonal and direct field movement • civilized, easy-to-remember function names • modular construction makes debugging/maintenance easy • generic data pointer makes customization easy • time and money functions • 32,000 + fields • error checking aids program development • clear and concise documentation • 800 support and bulletin board.

30-day guarantee: Following registration you'll get source code. No run-time license, no royalties.
\$179.00 C-scape (Lattice 3 + /Microsoft 3 +; others call) \$249.00 C-scape with Dan Bricklin's Demo Program
Please add \$3.00 for shipping. Massachusetts orders please add 5% sales tax.

#### Oakland Group, Inc.





Call today! 617-491-7311 or 800-233-3733

675 Massachusetts Avenue, Cambridge, MA 02139-3309

Circle no. 227 on reader service card.

#### MAC BUTTONS, AMIGA GADGETS (continued from page 47)

to identify the precise menu posting the event.

An Amiga program must also identify which menu and which item have been selected. The Amiga, though, reports only that an intuition event has occurred. The program must retrieve the input message with the system routine GetMsg (lines 205-208 in Listing Three) to determine exactly which of the intuition events requested in the window data structure has been detected. Once the type of event has been recovered (line 211), it can be compared against the desired types of events (lines 212-215) in the same way as the Macintosh program identifies events. The menu number and menu item (and the menu subitem, if applicable) are stored in a 16-bit field in the message data structure. The menu number is in bits 0-4 and can be isolated by masking off the 11 high-order bits (line 222). The menu item is in bits 5-10 (see lines 225-226).

The sample Macintosh program supports activities from all three menus. The Apple menu's desk accessories are handled by system routines. GetItem (lines 140 – 143 in Listing One) retrieves the name of the desk accessory that has been selected. Open-DeskAcc (lines 144-147) actually opens the program. Once the desk accessory has been opened, subsequent events in that program are posted to the event queue and detected as mouse-button down events in a system window (lines 104-105). In that case, the system routine SystemClick (lines 111-113) processes the event without further intervention from the application program. For the purposes of the sample program, only the Quit item is actually trapped from the File menu (lines 176-177); all other options simply return to the event loop. The CloseAndQuit routine (lines 187-191) frees the space used for text storage (TEDispose in lines 187-188), closes the window (CloseWindow in lines 189-190), and then returns to the Finder. The Edit menu, which is fully implemented with system routines, is discussed later in this article in the context of text editing.

The Amiga program also handles only the Quit item from its Project menu. Its CloseAndQuit routine (lines 229-240 in Listing Three) first removes any unprocessed console events with the system macro ABORTIO (this macro is found in the Amiga include files), then closes the console device (CloseDevice in lines 231-233), closes the window (CloseWindow in lines 234-236), and finally closes the custom screen (CloseScreen in lines 237-239) before returning to the operating system. The Edit menu is not implemented because the Amiga does not have system routines for text editing.

#### **Text Editing**

The environment for text editing is very, very different on the Macintosh and Amiga. Other writers have glossed over the Macintosh's text editing abilities (for example, see the September 1986 issue of *Byte* magazine<sup>3</sup>), but it is in this area that the difference between the Mac and Amiga system routines is glaringly apparent. The Macintosh's system routines for text editing are simple and elegant. The Amiga has nothing comparable; Amiga text I/O relies on low-level device communication.

To do text I/O, a Macintosh program allocates a text edit record with the system routine *TENew* (see lines 52–56 in Listing One). The text edit record is associated with a window, though the text stored in the text edit record can be much larger than what can be seen in the window at any given time. A call to *TEActivate* (lines 57–58) makes the text edit window active and draws the straight-line cursor as the text entry point. Repeated calls to *TEIdle* (line 61), usually within the program's event loop, ensure that the cursor blinks regularly.

When a text edit window is active, Macintosh programs generally assume that key-down events not associated with the command key represent text to be both displayed on the screen and stored in the text edit record. Whenever such an event is detected, the key pressed is stored in the event record. *TEKey* (lines 79–81 in Listing One) displays that character at the current cursor position, adjusting word wrap as necessary, and

#### FULL AT&T C++ for half the price of our competitors!

Guidelines announces its port of **version 1.1** of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. 'Object-oriented' means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2**. C++ will more than pay for itself in saved development time on your next project.

### C++

#### from GUIDELINES for the IBM PC: \$195

**Requires** IBM PC/XT/AT or compatible with 640K and a hard disk. **Note:** C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

#### Here is what you get for \$195:

- The full AT&T v1.1 C++ translator.
- Libraries for stream I/O and complex math.
- "The C++ Programming Language", the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- · Installation guide and documentation.
- 30 day money back guarantee.

#### To order:

send check or money order to:

GUIDELINES SOFTWARE P.O. Box 749 Orinda, CA 94563

To order with Visa or MC, phone (415) 254-9393. (CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.

Call or write for a free C++ information package.

Circle no. 351 on reader service card.

inserts the character into the text edit record in RAM.

Mouse-down events in active text edit windows signal that the user wishes to either change the position of the straight-line cursor or identify a block of text for editing activities. The Macintosh refers to blocks of code or the straight-line cursor as the "selection range." Setting the selection range requires calls to two system routines. GlobalToLocal (lines 115-116) is a graphics routine that takes the point where the mousedown event occurred, which is expressed in global screen coordinates, and translates it to the text edit window's local coordinate system. The transformed coordinates can then be passed to TEClick (lines 117-123), which actually sets the selection range.

The text editing functions listed in the Macintosh's standard Edit menu are each available as a single system routine that bases its operation on the current selection range. TECut (lines 156-159) deletes the current selection range from the screen, adjusting word wrap and the text edit record. The deleted text is stored in RAM in an area known as the "Clipboard." Cut, and any other operations that write to the Clipboard, erase the previous contents of the Clipboard. TECopy (lines 161-165) writes the current selection range to the Clipboard but does not affect the screen or text edit record. TEPaste (lines 166-169) inserts whatever it finds on the Clipboard at the current selection point (either at the straight-line cursor or after the selection range). Both the screen display and text edit record are adjusted. TEClear (lines 171-175) deletes the current selection range without affecting the Clipboard. These functions also provide "intelligent cut and paste," automatically adjusting spacing between words when text is either deleted, cut, or pasted.

There are two major ways to do text I/O on the Amiga—either via the console device, which processes keystrokes before passing them on to the system, or via the RAW device, which transmits unprocessed key codes. For simple text I/O, the console device is easier to use because it automatically manages special keys such as the backspace. To use the console device, an Amiga program must, as discussed

earlier, allocate two message ports one for input and one for output. These message ports are then incorporated into standard I/O request blocks, the data structures that are actually used for I/O. The subroutine CreateStdIO (lines 308-319, Listing Three) allocates memory for a standard I/O data structure and initializes the data structure with the address of its message port. Finally, the console device can be opened by the subroutine OpenConsole (lines 320-330). Note that the call to the system routine OpenDevice (line 327) returns a pointer to the device data structure in a field of one of the standard I/O request blocks. In other words, the device is linked to the I/O request blocks rather than the I/O request blocks being linked to the device. If the console is opened successfully, a solid block cursor appears in the upper left-hand corner of the window to which the console is attached. The cursor does not blink.

The system routine *SendIO*, which appears in the subroutine *QueueRead* (lines 332–337), issues a request for console input. *SendIO* uses data from the input standard I/O request block, including the type of operation to perform (line 332), the place where input should be stored (line 333), and the number of bytes to input (line 334).

The subroutine ConPutChar (lines 338-343) handles console output. Using the system routine DoIO (line 342), it displays a single character at the current cursor position and moves the cursor to the right. If the cursor is pushed past the right edge of the window, it drops to the leftmost position on the line below. ConPutChar does not do word wrap, nor does it store the text displayed in main memory. Subsequent calls to Queue-Read reuse the same storage space for input characters. As mentioned before, the Amiga also has no system routines for text editing. Therefore, although the backspace key can erase whatever character is displayed to the left of the cursor, no other editing is possible. To implement the standard text editing functions, programmers must write their own routines to do word wrap, manipulate the selection range, store text in main memory (or on disk if applicable), manage a clipboard, and do the editing operations.

#### The Bottom Line

It might be unfair to base an evaluation of the system routines of the Macintosh and Amiga simply on the subset of the routines designed to manipulate the user interface. On the other hand, the programming strategies used to implement the standard user interfaces are similar to those reguired for other system operations. In general, the Macintosh routines isolate programmers from low-level tasks such as list manipulation and initialization of data structures (File Manager parameter blocks are notable exceptions). Although this reduces the burden placed on the programmer, it can decrease the programmer's flexibility. The Macintosh routines are also more complete in terms of their support for the documented user interface. The effect is again to reduce the burden placed on the programmer.

On the Amiga, the intuition library provides routines for the standard user interface. Although support for screen, windows, menus, and fonts is available, there is a great gap in terms of text editing. In other words, the Amiga does not provide system routines to fully implement its own standard interface recommendations. As someone who writes more programs that rely on text manipulation than on graphics, I believe that this is a serious deficiency. It is true that the Amiga performs some functions "automatically" for which a Macintosh program must include code (for example, moving and sizing windows). Nonetheless, the Macintosh does include system routines to handle those functions.

#### Notes

- 1. Caroline Rose et al., *Inside Macintosh* (Reading, Mass.: Addison-Wesley, 1985).
- 2. Robert J. Mical and Susan Deyl, Intuition: The Amiga User Interface (Commodore-Amiga Inc., 1985).
- 3. Adam Brooks Webber, "Amiga vs. Macintosh," *Byte* (September 1986): 249-257.

DDJ

#### (Listings begin on page 64.)

Vote for your favorite feature/article. Circle Reader Service **No. 5**.

## THE PROGRAMMER'S SHOP

helps save time, money and cut frustrations. Compare, evaluate, and find products.

#### RECENT DISCOVERY

dBXL by Word Tech - complete interpreter clone. Adds windowing. Quicksilver, LAN support. Non-copy protected.

PC\$ 129

#### AI-Expert System Devit

Arity System-incorporate with	C		
programs, rule & inheritance	MS	\$	279
Experteach - Powerful, no limit	on		
memory size. Samples	PC	\$	399
EXSYS	MS	\$	319
Insight 2 + - dB2, language.	MS	\$	389
Texas Instruments:			
PC Easy	PC	\$	439
Personal Consultant PLUS	PC	\$2	2599

#### Al-Lisp

Microsoft MuLisp 85	MS	\$	199
PC Scheme LISP - by TI			85
TLC LISP - classes, compiler.	MS	\$	225
TransLISP - learn fast	MS	\$	85
Others: IQ LISP (\$155), UNX L	ISP (	\$5	9),
IQC LISP (\$269), WALTZLIS	SP (\$	139	9)

#### Al-Prolog

			375
APT - Active Prolog Tutor - bui	ld		
applications interactively	PC	\$	65
ARITY Standard - full, 4 Meg			
Interpreter - debug, C, ASM	PC	\$	319
COMPILER/Interpreter-EXE	PC	\$	739
With Exp Sys, Screen - KIT	PC	\$	1129
LPA MacProlog Complete - inc	reme	nta	ıl
compiler and an interpreter N	AAC	\$	295
LPA MicroProlog - intro	MS		
LPA MicroProlog Proffull			
memory	MS	\$	349
Prolog-86 - Learn Fast, Standard	d,		
tutorials, samples	MS	\$	89
Prolog-86 Plus - Develop	MS		229
TURBO PROLOG by Borland	PC	\$	69

#### Al-Other

METHODS - SMALLTALK	has	
objects, windows,		69
Q'NIAL - Combines APL wit	h LISP.	
Source or binary.		359
Smalltalk/V-graphics	PC	\$ 89

#### Atari ST & Amiga

We carry full lines of Man: Lattice, & Metacomco.	х,	
Amiga - LINT by Gimpe	l Amiga	\$ 79
Cambridge LISP	Amiga	
Lattice C S	T, Amiga	\$ 139
Lattice Text Utilities	Amiga	
Megamax - tight, full		200

#### **FEATURES**

Tom Rettig's Library - adds 140 functions to dBASE III Plus for arrays, character and date control, screen, new logical expressions, number manipulations, and much more. Full source (in C, assembler, and dBASE), no royalties. Use with Clipper. PC \$ 89 Pdisk - Improve your PC's performance with disk cacheing, extend DOS with single command tree structure manipulation and file MOVE, and maintain your hard disk with

PC \$ 159

Advanced Backup/Restore.

#### We Go Out of Our Way to Serve Developers

Our technical support provides accurate information on the product categories you need to be more productive. And we recommend the products that are right for you. We offer unbiased advice, free literature, and guarantees based on our recommendations. Often we suggest products or approaches that you might not have thought of. We supply every product for developers of software on PC's and every significant product for other environments. Call one of our qualified representatives today. How could better development tools help you? Call us.

Our Ser	
Programmer's Referral List	Dealers Inquire
Compare Products	Newsletter
· Help find a Publisher	Rush Order
• Evaluation Literature FREE	Over 700 products
. RRS . 7 PM to 7 AM 617-826-46	

#### Basic

Basic Development System - for			
BASICA; Adds Renum, more	PC	\$	105
Basic Development Tools by			
Sterling Castle	PC	\$	89
Basic Windows by Syscom	PC	\$	95
BetterBASIC - all RAM, module	es		
Structure. Full BASICA	PC	\$	139
8087 Math Support	PC	\$	89
Run-time Module	PC	\$	229
Better Tools - for Better Basic	PC	\$	95
CADSAM FILE SYSTEM-full	MS	\$	69
GoodBas - maintain code	PC	\$	95
LPI Basic - MS compatible UN	XIV	\$	1100
Prof. Basic - Interactive, debug	PC		79
8087 Math Support	PC	\$	47
QuickBASIC V2.0-New interface	PC	\$	67
TRUE Basic - ANSI	PC		109
Run-time Module	PC	\$	459
	25kg 200	73	12 13 1

#### Cobol

Macintosh COBOL - full	MAC	\$	459
MBP - Lev. II, native	MS	\$	819
Microsoft Cobol Tools - xref, d	ebugg	ger	
w/source support. Xenix \$359	PC	\$	219
Professional COBOL			2695
Realia - very fast			819
Ryan McFarland COBOL	MS	\$	699
COBOL-8X	MS	\$	995
VS Workbench			3500

#### **Editors for Programming**

BRIEF Programmer's Editor - u	ndo,		
windows, reconfigure	PC	1	Call
EMACS by UniPress - powerful	,		
multifile, windows Source:	929	\$	299
Epsilon - like EMACS, full			
C-like language for macros.	PC	\$	155
KEDIT - like XEDIT			109
Lattice Screen Editor - multiwin	dow,		
multitasking Amiga \$ 89	MS	\$	109
PC/VI - Custom Software new v	ersio	n	
fast	MS	\$	129
Personal REXX	PC	\$	115
PMATE - power, multitask	PC		
SPF/PC - fast, virtual memory	PC	\$	139
XTC - multitasking	PC	\$	85
	The same of	000	

#### C Libraries-Communications

Asynch by Blaise	PC	\$	135
Greenleaf Comm Lib.	PC	\$	149
Multi-Comm - add multitasking.	use		
w/Multi-C			149
Software Horizons pack 3		-	119

#### RECENT DISCOVERY

C86 PLUS - New version has 70% faster execution speed, tight code full ANSI library. Support for source level debugger, ROMable. Library C source. Ask about benchmarks

MS \$459

#### C Language-Compilers

AZTEC C86 - Commercial	PC	\$499
C86 by CI - 8087, reliable	MS	\$299
Datalight C - fast compile, good	code,	
4 models, Lattice compatible,	Lib	
source. Dev'rs Kit	PC	\$ 77
HOT C - new, intriguing	PC	\$ 85
Lattice C - from Lattice	MS	\$299
Mark Williams - w/debugger	MS	\$369
Microsoft C 4.0- Codeview	MS	\$279
Wizard C - Lattice C compatible	,	
full sys. III, lint, fast.	MS	\$359

#### C Language-Interpreters

by Gimpel - full K & R MS S	\$229
ANT C - Source debug,	***
to Run-3 seconds, .OBJs MS S	\$389
ctive C by IMPACC Assoc.	
g. PC S	\$225
ucing C-self paced tutorial PC	\$105
Professional MS S	\$179
Lite - improved MS	\$ 97

#### C Libraries-General

Blackstar C Function Library	PC	\$	79
C Essentials - 200 functions	PC	\$	83
C Food by Lattice-ask for source	MS	\$1	09
C Scientific Subroutines - Peerless	MS	\$1	35
C Tools Plus (1 & 2)	PC	\$1	35
C Utilities by Essential - Compre	hensi	ve	
screen graphics, strings, source.			37
C Worthy Library - Complete, m	achin	e	
independent	MS	\$2	95
Entelekon C Function Library	PC	\$1	19
Entelekon Superfonts for C	PC		
Greenleaf Functions-portable, A	SM	\$1	39
PforCe by Phoenix - objects	PC	\$2	99

#### C Libraries-Files

FILES: C Index by Trio - full B		
Tree, vary length field, multi cor	npiler	
/File is object only		\$ 89
/Plus is full source	MS	\$349
CBTREE - Source, no royalties	MS	\$ 99
CTree by Faircom - no royalties	MS	\$329
dbQUERY - ad Loc, SQL - based	MS	\$159
dbVISTA - full indexing, plus or	otiona	1
record types, pointers, Network	<b>.</b>	
Object only - MS C, LAT,	C86	\$159
Source - Single user		\$429
Source - Multiuser	MS	\$849
dBASE Tools for C	PC	\$ 65
dbc Isam by Lattice	MS	\$199
dBx - translator	MS	\$319
w/source	MS	\$519

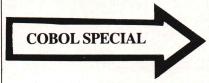
#### FEATURE

C Trainer by Catalytix - full C interpreter plus Prentice-Hall text covers introductory to advanced topics, proposed ANSI standard. Integrate with own editor. Extensive error detection.

PC \$115

We support MSDOS (not just compatibles), PCDOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.

provides complete information, advice, guarantees and every product for Microcomputer Programming.



Call our COBOL Specialist to discuss compilers & support products to see what fits your background & inter-

PC \$139

MS \$ 95

PC \$149

PC \$115

PC \$349

\$ 99

\$199

\$369

\$295

\$129 PC

\$159 PC

MS \$189

PC \$129

PC \$107

PC \$139

PC \$249

PC \$119

PC \$249

PC \$135

PC \$ 329

\$199

\$ 89

PC \$209

PC \$ 69

PC \$217 \$295

PC \$239

\$600

\$179 MS \$ 95

C Support-Systems

Basic-C Library by C Source

C ToolSet - DIFF, xref, source

Quickshell - script compiler

ESSENTIAL GRAPHICS - fast,

Multi-Windows - use w/ Multi-C

Windows for Data - validation

Advanced Trace-86 by Morgan Modify ASM code on fly

CODESMITH - visual, modify and rewrite Assembler

C SPRITE - data structures

DSD87 - by Soft Advances

Periscope II - symbolic, "Reset Box," 2 Screen

Showcase by Test Software

Software Source by Atron -Lattice, MS C, Pascal, Windows single step, 2 screen, log file. MS \$115

w/Breakswitch FEATURE

detailed specs.

Pfix-86 Plus Symbolic Debugger by Phoenix - windows

Pascal-2 - Perhaps tightest compiler for MSDOS. Mainframe background & power. MS compatible. Complete

environment. Turbo translator. Get

Periscope I - own 16K

GraphiC - mono version GraphiC - new color version

dBASE Graphics for C

Greenleaf Data Window

Vitamin C - screen I/O

ZView - screen generator

Windows for C - fast

Debuggers

Curses by Lattice

w/source

Lattice Text Utilities Multi-C - multitasking

C Sharp - well supported. Source, realtime, tasks, state system

The HAMMER by OES Systems PC

PC LINT-Checker. Amiga \$89 MS \$107 SECURITY LIB-add encrypt to MS C C86 programs. Source \$229

C-Screens. Windows. Graphics

C Power Windows by Entelekon PC \$119

Order before 1/31/87 and mention this ad for these special prices.

By MicroFocus	List	Normal	SPECIAL
Level II	1500	1350	589
Professional	3500	2695	1795
with Sourcewriter —			
Professional	5000	4554	2499
VS Workbench	6000	5349	4289
Microsoft COBOL	700	499	459
Ryan McFarland COBOL	950	699	595
COBOL-8X	1250	995	785

#### Fortran & Supporting

MS	\$429
d	
MS	\$ 59
MAC	\$229
MS	\$209
PC	\$119
MS	\$389
MS	\$149
MS	\$259
ell PC	\$ 59
	MS MAC MS PC MS PC

#### Multilanguage Support

BTRIEVE ISAM	MS	\$199
BTRIEVE/N-multiuser	MS	\$469
CODESIFTER - Profiler.	MS	\$ 99
HALO Graphics - 115 + devices		
Animation, engineering, busine		
Any MS language, Lattice, C86		\$217
Informix - by RDS		\$679

Informix - by RDS	PC	\$679	
Informix 4GL - application builder	PC	\$849	
PANEL - Create screen with editor	,		
generates code. Full data validation	n,		
no royalties. Xenix \$539,	MS	\$219	
PolyLibrarian by Polytron	MS	\$ 85	
PVCS Version Control	MS	\$329	
QMake by Quilt Co.	MS	\$ 89	

Rtrieve - Xtrieve option		\$11	9
Screen Sculptor - slick, thorough	,		
fast, BASIC, PASCAL.		\$ 9	9
Xtrieve - organize database	MS	\$19	9
ZAP Communications - VT 100,			
TEK 4010 emulation, file xfer.	PC	\$ 8	9

#### Pascal and Supporting

ALICE - learn Pascal, Turbo		
compatible, interpreter	PC	\$ 79
Exec - Chain Programs	MS	\$ 85
MetaWINDOW - graphics toolki	t PC	\$135
Microsoft PASCAL - faster	MS	\$199
MICROTEC PASCAL - 5 memo	ry mo	odels,
"Iterators", 65 bit 8087 strings	MS	\$665
Pascal Pac with Tidy - formatter,		
utilities	PC	\$ 69
Pascal Tools - strings, screen	PC	\$109
Pascal Tools 2 - by Blaise	MS	\$ 85
Pfas - Portable Isam	MS	\$185
TurboHALO - 150 routines, IBM		
EGA, Hercules, more	PC	\$ 85
USCD Pascal - native code	MS	\$ 69

#### RECENT DISCOVERY

TransLISP PLUS - with C INTERFACE, 400 + COMMON LISP functions. Optional UNLIMITED Runtime PLUS for MSDOS \$ 179

Other Languages			
APL*PLUS/PC	PC	\$	469
Artek ADA Compiler - DODs	tandar	d	
minus multitasking	PC	\$	469
CLIPPER-dBASE Compiler	MS	\$	429
Correct FORTH - ROMable	PC	\$	80
ED/ASM - 86 by Oliver	PC	\$	85
Lattice RPG II Compiler	PC	\$	719
MacASM - fast	MAC	\$	99
MasterForth - Forth '83 MAC	or PC	\$	109
Microsoft MASM - faster	MS	\$	98
Modula 2 by Volition Systems	MS	\$	250
Modula-2/86 Compiler by Log	itech		
w/8087 (\$ 99), 512K (\$145)		\$	62
Pasm - by Phoenix	MS	\$	149
SNOBOL4 + - great for string	s MS	\$	85
Turbo Edit/ASM - by Speedward		\$	85

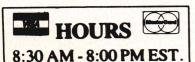
#### Xenix-86 & Supporting Basic - by Microsoft 249 Cobol - by Microsoft 659 Fortran or Pascal - by Microsoft 489

#### MicroFocus Lev. II Compact COBOL\$ 795 Xenix Complete Development System\$1049

Other Products			
386 Assembler/Linker	PC	\$	459
ASMLIB - 170 + routines	PC	\$	135
BSW Make - like UNIX make	MS	\$	85
Dan Bricklin's Demo Program	PC	\$	65
dBrief - Customize BRIEF for d	BAS	E	
development. with BRIEF \$275	. PC	\$	95
H Test/H Format - XT Fix	PC	\$	89
Interactive Easyflow-HavenTree	PC	\$	129
Link & Locate - tools to work wi	th		
Intel and Tektronix projects.	MS	\$	329
LMK - like UNIX make	MS	\$	149
Microsoft Windows	PC	\$	75
Microsoft Windows Software			
Development Kit	PC	\$	349
MKS Toolkit - Unix, vi, awk	PC	\$	119
Opt Tech Sort - sort, merge	MS	\$	119
PMaker - by Phoenix	PC	\$	99
Polymake by Polytron	MS	-	
PS MAKE by UniPress	MS	\$	79
SECRET DISK by Lattice	PC	\$	95
SET:SCIL - manager revisions	PC	\$	299
Shrink/Shrinkem - put more file			
on disk with spacemaker	PC	\$	150
SoftEst - Manage projects.	MS	\$	
Synergy-Create user interfaces	MS		375
Texsys - control source	MS	\$	89
Visible Computer: 8088 - Simul	ates		
demos or any .exe. com, Debu			(0
350 pg. tutorial, unprotected	PC	4	69

Note: All prices subject to change without notice. Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3' laptop now available, plus 200 others. UPS surface shipping add \$3/item.

#### Call for a catalog, literature, advice and service you can trust



800-421-8006

THE PROGRAMMER'S SHOP™ 128-D Rockland Street, Hanover, MA 02339 Mass: 800-442-8070 or 617-826-7531 11/86 "I like your straightforward, open evaluations, comments and selection."

> Chris Chapman Practical Solutions Software

# Is \$19.97 too much to spend to avoid a million dollar mistake?

#### **Business Software**

magazine is a valuable resource that helps you squeeze every penny's worth of potential out of your computer system. It is written for the business person and decision maker who wants practical, "hands-on" software solutions to real problems, written in a straightforward, jargon-free language. Every month, Business Software brings you: case studies of business computer applications; company profiles, including who is using what software and why; software tutorials that explain what the manual left out; information on the latest products, as well as the longtime favorites, all from a business perspective.

To sul	oscribe	e, p	leas	e fill	out	this
coupc	on and	ret	urn	it to	):	

#### **Business Software**,

P.O. Box 27975, San Diego, CA 92128

5an Diego, CA 72126	
Name	1 - 1 - 1
Address	
City	
State Zip	
Bill my ☐ VISA ☐M/C	
☐ Bill me later ☐ \$19.97	
Card No.	
Expiration date	
Signature	
Please allow 6-9 weeks for address change to take	ag 2021

#### 68000 MINI FORTH

#### Listing One (Text begins on page 22.)

```
; Listing One
; FLINT's system macros
MACRO definition file for FLINT as implemented on the SAGE II microcomputer.
determines the environment (if any) into which the exit is made
               . MACRO
                                XSTOP
                .WORD
                                4EF9H
                . WORD
                                                    exit to PROM monitor/debugger
                .WORD
                                                    gets a character from the terminal and puts its ASCII value in the lower byte of D0
                                TGET
                . MACRO
                . WORD
                                 4EB9H
                                                    PROM BIOS call
                . WORD
                . WORD
                                 0008H
                . MACRO
                                                    sends the character whose ASCII value is in the lower byte of D0 to the
                                TPUT
                                                    terminal
                .WORD
                                 4EB9H
                .WORD
                                                    PROM BIOS call
                                 0014H
                .ENDM
                                                                  loads 1024 bytes from the disk block
numbered by the value in CBLOCK into
a buffer whose address is DISKBUF
                . MACRO
                                 GETBLOCK
                                A4, - (A7)
CBLOCK, - (A7)
DISKBUF, A0
A0, - (A7)
#1024, - (A7)
#1, - (A7)
4EB9H
OOFFH
               MOVE.L
               MOVE.W
                                                                  push block #
               LEA
MOVE.L
                                                                  push buffer address
push length of transfer (1024 bytes)
push drive #
               MOVE. L
                MOVE.W
                .WORD
                                 OOFEH
                                                                  PROM BIOS call for diskread
               .WORD
MOVEA, L
                              (A7) + A4
                                                                  recover A4
                .ENDM
                . MACRO
                                 SAVBLOCK
                                                                  saves the contents of a 1024 byte buffer whose address is DISKBUF onto
                                                                  the disk block numbered by the value in CBLOCK
               MOVE.L
                                 A4,-(A7)
               MOVE.W
                                 CBLOCK, - (A7)
DISKBUF, A0
                                                                  push block #
               LEA
MOVE.L
                                                                  push buffer address
push length of transfer (1024 bytes)
push drive #
                                 A0, - (A7)
                                 #1024,-(A7)
#1,-(A7)
4EB9H
                MOVE. L
                . WORD
                 . WORD
                                 OOFEH
                                                                  PROM BIOS call for diskwrite
               .WORD
MOVEA.L
                              (A7) + A4
                                                                  recover A4
                .ENDM
```

**End Listing One** 

#### **Listing Two**

; Listing Two
; The FLINT Interpreter

.NOLIST
.NOSYMTABLE
.NOMACROLIST
.NOPATCHLIST
.INCLUDE MACROS.TEXT
.LIST

```
We note the effect that the execution of a word has on the stack by a conventional shorthand. Before and after lists of the relevant stack parameters are given in "rightmost is topmost" order separated by "-->". An address is denoted by "a", an integer by "n", and a flag by "fl". Parameters in parentheses may or may not be present.

word action stack effects

TOKEN gets token from input buffer

SEARCH searches dictionary for current token pushes its code address and true flag or else false flag

EXECUTE executes routine whose code address a --> is on stack

NUMBER determines if a token whose address is popped from stack represents a number if so pushes value and true flag or else pushes false flag
```

```
sends ? to terminal
     WHAZZAT
                            makes a dictionary header for the next
token in the input stream (the name of
a word being defined).
     HEADER
                            calls HEADER and then sets the compile mode (colon) flag
                            writes code for JSR in the dictionary and calls "," (which furnishes the operand for JSR)
      COMPILE
                                                                                          a -->
     CODE an "immediate"
                            ate" word (executed even in
compile mode) which sets the base to hex
and sets the code submode flag
                            takes a number from the stack and writes it in the dictionary directly
                            takes a number from the stack and generates code which when executed will push the number back on the stack
      LITERAL
                            closes the current definition by writing an RTS in the dictionary and clearing the
                            compile and code flags.
     PROMPT
                            sends prompt (ok) to terminal
            gets a line from the input device and places it in the line buffer
  control structure of the prompt and input code PROMPT
                                  outer interpreter loop
                                            SEARCH
                                            if found
                                                EXECUTE
                                                           (execute mode)
                                                STKCHK
                                                COMPILE
                                                           (compile mode)
                                            else
NUMBER
                                                if fail
                                                       WHAZZAT
                                                       COMMA (code mode)
                                                       LITERAL
                                                                       (compile mode)
                                                       return
                                                                     (execute mode)
  register usage
                line buffer pointer
                                                                 I/O port
"scratch
                dictionary pointer
parameter stack pointer
return stack pointer
      A5
                                                          D1
                                                                        registers"
.ABSOLUTE
               .PROC
                                FLINT
                                                          ; set up
               BRA
                                START
                                                                          system buffers,
TERMBUF
               . BLOCK
                                84,32
                BLOCK
ASCII
                                1024,32
"INTERACTIVE "
                .BYTE
                BYTE
                                32
                                                                          system stacks,
               . BLOCK
                                80,0
RTNSTK
PARMSTK
UNDRFLW
                .BLOCK
.WORD
                                256,0
DICT
CBLOCK
                .WORD
                                LAST
                                                                          system variables,
                                0050H
BASE
                .WORD
                                10
                                                                          system flags,
                .BYTE
FCOLON
FIMMED
FCODE
                .BYTE
FNEG
                .BYTE
FINT
                BYTE
BLANK
                . BLOCK
ZERO
                BLOCK
                                3,32
                                RTNSTK+80, A7
                                                           : initialize pointers
START
               LEA
                                 PARMSTK+256, A6
               MOVEA.W
CLR.L
                             BUFPNT, A5
FCOLON
FINT
                                                             initialize flags
                CLR.B
                                                             load boot screen
                JSR
JSR
                                 TOAD
                                 WHICHBUF
                                                              select input buffer
RESTART
                                                             get next token
                TSR
                                 TOKEN
MAIN
                                                             push dictionary pointer
look for token
if found then
                MOVE . W
                                DICT, - (A6)
SEARCH
                JSR
TST.W
                                 (A6)+
                                 TSTNUM
#7,FIMMED
               BEQ
BCLR
                                                                      if immediate flag off then
                                                                                     (continued on next page)
```

**COMBINE THE RAW POWER OF FORTH** WITH THE CONVENIENCE OF CONVENTIONAL LANGUAGES

## FOR

Why HS/FORTH? Not for speed alone, although it is twice as fast as other full memory Forths, with near assembly language performance when optimized. Not even because it gives MANY more functions per byte than any other Forth. Not because you can run all DOS commands plus COM and EXE programs from within HS/FORTH. Not because you can single step, trace, decompile & dissassemble. Not for the complete syntax checking 8086/ 8087/80186 assembler & optimizer. Nor for the fast 9 digit software floating point or lightning 18 digit 8087 math pack. Not for the half megabyte LINEAR address space for quick access arrays. Not for complete music, sound effects & graphics support. Nor the efficient string functions. Not for unrivaled disk flexibility - including traditional Forth screens (sectored or in files) or free format files, all with full screen editors. Not even because I/O is as easy, but far more powerful, than even Basic. Just redirect the character input and/ or output stream anywhere - display, keyboard, printer or com port, file, or even a memory buffer. You could even transfer control of your entire computer to a terminal thousands of miles away with a simple >COM <COM pair. Even though a few of these reasons might be sufficient, the real reason is that we don't avoid the objections to Forth — WE ELIMINATE THEM!

Public domain products may be cheap; but your time isn't. Don't shortchange yourself. Use the best. Use it now!

HS/FORTH, complete system: \$395. with "FORTH: A Text & Reference" by Kelly and Spies, Prentice-Hall and "The HS/FORTH Supplement" by Kelly and Callahan



Mastercard Mastercard



#### HARVARD SOFTWORKS

PO BOX 69 SPRINGBORO, OH 45066 (513) 748-0390

#### Byte Magazine called it.



#### The SBI80 Computer/Controller

Featured on the cover of Byte, Sept. 1985. the SB180 lets CP/M users upgrade to a fast, 4" x 7½" single board system.

#### • 6MHz 64180 CPU

(Z80 instruction superset), 256K RAM, 8K Monitor ROM with device test, disk format, read/write.

 Mini/Micro Floppy Controller (1-4 drives, Single/Double Density, 1-2 sided, 40/77/80 track 3½", 5½" and 8" drives).

Measures 4" x 7½", with mounting holes

**One Centronics Printer Port** Two RS232C Serial Ports

(75-19,200 baud with console port auto-baud rate select).

Power Supply Requirements +5V +/-5% @500 mA +12V +/- 20% @40mA ZCPR3 (CP/M 2.2/3 compatible)

Multiple disk formats supported

Menu-based system customization

#### SB180-1

SB180 computer board w/256K bytes RAM and ROM monitor \$299.00

#### SB180-1-20

same as above w/ZCPR3, ZRDOS and BIOS source.....\$399.00

-Quantity discounts available-

#### NEW

#### COMM180-M-S

optional peripheral board adds 1200 bps modem and SCSI hard disk interface.

TO ORDER CALL TOLL FREE 1-800-635-3355

TELEX 643331

For technical assistance or to request a data sheet, call: 1-203-871-6170



Micromint, Inc. 4 Park Street Vernon, CT 06066

#### 68000 MINI FORTH

#### Listing Two (Listing continued, text begins on page 22.)

	BNE	GODO	是,我们就不是一个一个一个一个一个一个一个一个一个一个一个一个一个一个一个一个一个一个一个
	TST.B BEQ	FCOLON GODO	; if in compile mode
	JSR	COMPILE	; compile
gono	BRA	MAIN	; else
GODO	JSR JSR	EXECUTE	; execute word ; and check underflow
	BRA	MAIN	; else
TSTNUM	MOVE.W	A5,-(A6)	; push token buffer address
	JSR TST.W	NUMBER (A6)+	; is token a number ?? ; if not
	BNE	TSTCODE	; if not ;
	JSR BRA	WHAZZAT MAIN	; ? whazzat ? ; else
TSTCODE	TST.B BEQ	FCODE TSTLIT	; if code flag on ;
	JSR	COMMA	; write code in dictionary
	BRA	MAIN	; else
TSTLIT	TST.B	FCOLON	; if in compile mode
	BEQ JSR	MAIN LITERAL	; compile number as literal
	BRA	MAIN	,
WHICHBUF T	ST.B	FINT	
	BNE LEA	GOLINE	; if not in interactive mode
	RTS	DISKBUF, A4	; get input from disk buffer ; else
GOLINE	JSR	TIME	; fill line buffer from terminal
GOLINE	RTS	LINE	; fill line buffer from terminal ;
LINE	JSR	PROMPT	; prompt
	MOVEQ	#76,D1	; prompt
CLEANUP	LEA MOVE.L	TERMBUF, A4 BLANK, 0 (A4, D1)	; ; clear line buffer
Chimator	SUBQ.B	#4,D1	; Clear line builer
	BGE CLR.L	CLEANUP D1	; clear character count
INCHAR	TGET		; get next character and
	CMPI.B BEQ	#13,D0 EXIT	; do while not CR
	CMPI.B	#8,D0	; if character is backspace
	JSR	INBUF RUBOUT	; rubout previous char
	BRA	INCHAR	; else
INBUF	MOVE.B ADDQ.B	D0,0(A4,D1) #1,D1	; copy it into buffer ; increment count
	TPUT		; echo to terminal
EXIT	BRA MOVE.B	INCHAR D0,1(A4,D1)	; imbed CR in buffer
	RTS		
RUBOUT	TST.B	D1	; if count > 0 then
	BLE	RRET	
	TPUT SUBQ.B	#1,D1	; echo backspace ; decrement count
	MOVEQ MOVE.B	#32,D0	; erase previous character
	TPUT	D0,0(A4,D1)	; in buffer and ; on terminal
	MOVEQ	#8,D0	
RRET	RTS		; return
STKCHK	LEA	UNDRFLW, A0	
	CMPA.W	A0, A6	; if top is below bottom
	BLE JSR	OKSTK STKU	; underflow exception
OKSTK	RTS		;
NULLWORD .	BLOCK	6,0	; DICTIONARY
	.BYTE	4	
	.ASCII	"CRL"	; CRLF
CRLF	.WORD MOVEQ	NULLWORD #13,D0	
	TPUT MOVEQ	#10,D0	; send CR
	TPUT	410,D0	; send LF
	RTS		; return
	.BYTE	6	
	.ASCII	"PRO" CRLF-6	; PROMPT
PROMPT	JSR	CRLF	; send CR LF
	MOVEQ TPUT	#111,D0	; send "o"
	MOVEQ TPUT	#107,D0	
	MOVEQ	#32,D0	; send "k"
	TPUT		; send space ; return
	.BYTE	7 "WHA"	; WHAZZAT
WHAZZAT	.WORD	PROMPT-6	
.muezni	JSR MOVE.W	CRLF A5,-(A6)	; send CR LF ; push token address
	ADDI.W MOVE.B	#1, (A6) (A5), - (A6)	
	CLR.B	-(A6)	; push character count ; pad it to "word" length
	JSR BRA	TYPE RSET	; type offending token
STKU	JSR MOVEQ	CRLF	; (underflow entry point)
	TPUT	#42,D0	; ; send *

```
MOVEQ
                                  #63.D0
                                                                  send ?
                TPUT
                                  PARMSTK+256 A6
                                                               ; reset stack pointer
                LEA
                                                               ; initialize flags
; set interactive mode
; get new line
                CLR.L
TAS
                                  FCOLON
                                  FINT
                JSR
                                  CR
                 .BYTE
                 ASCII
WORD
                                   "TOK"
                                                                      TOKEN
                                  WHAZZAT-6
                                                                  clear token buffer
TOKEN
                CLR. L
                                   (A5)
                                  D1
(A4)+,D0
#32,D0
                 CLR I
                                                                  clear count
                MOVE . B
                                                                  getcharacter until space
GETCHAR
                CMPI.B
                                  EXITGET
D0,1(A5,D1)
#1,D1
                MOVE . B
                                                                      place in token buffer increment count
                ADDQ. B
                                  GETCHAR
D1, (A5)
                BRA
MOVE.B
                                                                  put count in 1st byte of buffer if count is 0 repeat
EXITGET
                 BEO
                                   GETCHAR
                                                                  return
                 BYTE
                                   129
                                                                       (CR)
                 . WORD
                                   TOKEN-6
                  .WORD
                                                               ; reset system stack
; restart
                                  RTNSTK+80, A7
RESTART
CR
                 JMP
                                   6
"SEA"
                  BYTE
                 . ASCII
                  WORD
                                   CR-6
                                                                  put token "stem" in D1
                                    (A5),D1
SEARCH
                 MOVEA.W
                                A6, A0
                                                                   use A0 as search pointer
                                   (A0)
COMPARE
                 TST. W
                                (A0)
(A0),A0
NOFIND
(A0),D1
FIND
#31,D1
(A0),D1
FINDIMM
#31,D1
4(A0),A0
                                                                  DO
                                                                       get address of next word
if nullword, exit NOFIND
compare word to candidate
if found, exit FIND
set precedence bit
compare to "immediate" candidate
                 MOVEA.W
                 BEO
                 CMP. L
                 BEQ
BCHG
                 CMP . T.
                                                                       if found, exit FINDIMM reset precedence bit get link address
                 BCHG
                 LEA
                                   4 (A0) . A0
                 BRA
                                   COMPARE
FIMMED
                                                                  T.OOP
                                                                   set immediate flag
get code address
FINDIMM
                                   6 (A0) . A0
                 T.F.A
                                                                   push it
                 MOVE.W
MOVE.W
                                   A0, (A6)
#-1,-(A6)
                                                                   push success flag
                 RTS
                                                                   push fail flag
                 MOVE.W
                                   A0, (A6)
NOFIND
                 RTS
                  BYTE
                                   "NUM"
                                                                       NUMBER
                  ASCII
                  WORD
                                   SEARCH-6
 NUMBER
                                D2
(A6)+,A0
                                                                  get token address
get digit count
DO
                                                                   clear conversion register
                 MOVEA.W
                                    (A0)+,D1
                 MOVE B
                                                                       strip ASCII prefix
if digit too small, FAIL
if digit > 9
                                                                       get next digit
                 MOVE . B
 NXTDIG
                  SUBI.B
                                    #48.D0
                 BLT
CMP.W
                                   FAIL
#10,D0
                                                                               adjust for "odd" values
                                    CMPBASE
                  BLT
                                    #7,D0
#10,D0
                                                                               and test again
                  SURT P
                  BLT
                                    FAIL
                                                                       if base < digit
 CMPBASE
                  CMP.W
                                    BASE, DO
                                                                               FATL
                                    BASE, D2
                                                                        multiply current value by base
                  MULU
                  SWAP
                                    D2
                                   D2
FAIL
                                                                        if overflow
                  TST.W
                  BNE
                                   D2
D0, D2
                  SWAP
                                                                   add current digit
if overflow, FAIL
decrement count
UNTIL no digits remain
                  ADD.W
                  BCS
                                    FAIL
#1,D1
                  SUBO. B
                  BNE
MOVE.W
                                    NXTDIG
                                                                   push number
push success flag
                                    D2, -(A6)
#-1, -(A6)
                  MOVE. W
                  RTS
CLR.W
                                                                 ; push fail flag
                                     -(A6)
 FAIL
                   .BYTE
                   ASCII
                                    "EXE"
                                                                        EXECUTE
                                    NUMBER-6
                                                                   pop code address
                  MOVEA.W
 EXECUTE
                                 (A6)+,A0
                   TSR
                                                                    execute
                   BYTE
                   .ASCII
                                     IICOMII
                                                                        COMPTLE
                                                                   compile "JSR"
                                     #04EB8H, (A5) +
  COMPTLE
                  MOVE.W
                                                                 ; compile code address
                                     COMMA
                   BYTE
                                     6
"HEA"
                   ASCII.WORD
                                                                        HEADER
                                     COMPILE-6
DICT, 4 (A5)
A5, DICT
                                                                   link header to dictionary update DICT
                   MOVE . W
  HEADER
                   MOVE . W
                                                                  ; move pointer to code field
                                     6 (A5), A5
                    BYTE
                                     9
"IMM"
                                                                         IMMEDIATE
                                     HEADER-6
                    WORD
                                                                                              (continued on next page)
```

# Complete C Programs in Half the Time, with Instant-C

You can create programs much faster with Instant-C than with conventional programming tools. How? Because Instant-C is a high-performance interpreter, there are **no compile or link delays.** Change your program, then test it immediately. No matter how large your program, the turnaround time is just seconds.

"Instant-C means instant gratification."—*PC Magazine*, **Editor's Choice** for best C interpreter.

Powerful **source-level debugging** saves your time. Conditional breakpoints, single-stepping by statement, source code backtraces, data monitoring, and many other debugging features make it easy to wipe out bugs quickly. Direct execution of any statement or function makes testing a breeze.

"The resulting debugging and testing capabilities are fantastic and the detailed trace/debug/display commands make it easy." — *The C. Journal* 

Instant-C checks pointer references for reasonableness, and checks that array indexes are within declared bounds. This **run-time checking** stops your program as soon as errors occur, for easiest debugging.

Not only does Instant-C help you quickly change, test, check and debug your code, but it runs your program fast enough for real-time applications.

"It is much faster than any of the other products mentioned and was the only one able to complete the standard SIEVE in a reasonable time. Clearly, this high speed allows much more complex problems to be attacked with Instant-C than with any of the other products discussed."—Computer Language

Immediate feedback and precise diagnostics make Instant-C great for learning C. Full K&R and the ability to **link compiled object code and libraries** (Lattice and Microsoft) makes Instant-C compatible with your existing programs.

Instant-C makes all parts of the programming task as fast as possible.

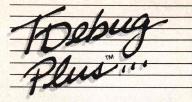
"Clearly, Instant-C is the performance champion."—PC Tech Journal

Version 2 works with MS-DOS and PC-DOS, and has a full 31 day **money back guarantee.** Instant-C is only \$495. Order today! Call or write for full information.

Rational Systems, Inc.

P.O. Box 480 Natick, MA 01760 (617) 653-6194

#### TURBO PROGRAMMERS-



#### ... CUTS DEBUGGING FRUSTRATION.

TDebug PLUS is a **new**, interactive symbolic debugger that integrates with Turbo Pascal to let you:

- Examine and change variables at runtime using symbolic names – including records, pointers, arrays, and local variables;
- Trace and set breakpoints using procedure names or source statements;
- View source code while debugging;
- Use Turbo Pascal editor and DOS DEBUG commands

TDebugPLUS also includes a special MAP file generation mode fully compatible with external debuggers such as Periscope, Atron, Symdeb, and others — even on programs written with Turbo EXTENDER

An expanded, supported version of the acclaimed public domain program TDEBUG, the TDebugPLUS package includes one DSDD disk, complete source code, a reference card, and an 80-page printed manual. 256K of memory required. Simplify debugging! \$60 COMPLETE.

#### TURBO EXTENDER TM

Turbo EXTENDER provides you the following powerful tools to break the 64K barrier:

- Large Code Model allows programs to use all 640K without overlays or chaining, while allowing you to convert existing programs with minimal effort; makes EXE files;
- Make Facility offers separate compilation eliminating the need for you to recompile unchanged modules;
- Large Data Arrays automatically manages data arrays up to 30 megabytes as well as any arrays in expanded memory (EMS);
- Additional Turbo EXTENDER tools include Overlay Analyst, Disk Cache, Pascal Encryptor, Shell File Generator, and File Browser.

The Turbo EXTENDER package includes two DSDD disks, complete source code, and a 150-page printed manual. Order now! \$85 COMPLETE.

#### TURBOPOWER UTILITIES TM

"If you own Turbo Pascal, you should own TurboPower Programmers Utilities, that's all there is to it." Bruce Webster, BYTE Magazine

TurboPower Utilities offers nine powerful programs: Program Structure Analyzer, Execution Timer, Execution Profiler, Pretty Printer, Command Repeater, Pattern Replacer, Difference Finder, File Finder, and Super Directory.

The TurboPower Utilities package includes three DSDD disks, reference card, and manual. \$95 with source code; \$55 executable only.

#### ORDER DIRECT TODAY!

- MC/VISA Call Toll Free 7 days a week 800-538-8157 x830 (US) 800-672-3470 x830 (CA)
- Limited Time Offer! Buy two or more TurboPower products and save 15%!
- Satisfaction Guaranteed or your money back within 30 days.

For Brochures, Dealer or other Information, PO, COD—call or write:



3109 Scotts Valley Dr., #122 Scotts Valley, CA 95066 (408) 438-8608 M-F 9AM-5PM PST

The above TurboPower products require Turbo Pascal 3.0 (standard, 8087, or BCD) and PC-DOS 2.X or 3.X, and run on the IBM PC/XT/AT and compatibles.

Circle no. 207 on reader service card.

#### 68000 MINI FORTH

#### Listing Two (Listing continued, text begins on page 22.)

IMMED	MOVEA.W TAS RTS	DICT,A0 (A0)	; get address of most recent word ; set precedence bit
COLON	.BYTE .ASCII .WORD .WORD JSR JSR TAS	1 ":" 0 IMMED-6 TOKEN HEADER FCOLON	; ":" ; get token ; make header ; set colon flag
CODE	BYTE .ASCII .WORD MOVE.W TAS RTS	132 "COD" COLON-6 #16, BASE FCODE	; CODE ; change BASE to hex ; set code flag
SEMI	.BYTE .ASCII .WORD .WORD MOVE.W CLR.B CLR.B MOVE.W	129 "," 0 CODE-6 \$10,BASE FCOLON FCODE \$04E75H, (A5)+	; ";" ; change BASE to decimal ; clear colon flag ; clear code flag ; compile "RTS"
сомма	.BYTE .ASCII .WORD .WORD MOVE.W RTS	1 "," 0 SEMI-6 (A6)+,(A5)+	; "," ; pop number to dictionary
LITERAL	.BYTE .ASCII .WORD MOVE.W JSR RTS	"LIT" COMMA-6	<pre>; LITERAL ; compile literal code ; compile constant</pre>
INTRACTV T	.BYTE .ASCII .WORD 'AS RTS	11 "INT" LITERAL-6 FINT	; INTERACTIVE ; set interactive mode
BASECODE L	.BYTE .ASCII .WORD EA MOVE.W RTS	4 "BAS" INTRACTV-6 BASE, A0 A0, - (A6)	; BASE ; push BASE address
CBLCODE	.BYTE .ASCII .WORD LEA MOVE.W RTS	6 "CBL" BASECODE-6 CBLOCK,A0 A0,-(A6)	; CBLOCK ; push CBLOCK address
EDBCODE	.BYTE .ASCII .WORD LEA MOVE.W RTS	5 "EDB" CBLCODE-6 DISKBUF,A0 A0,-(A6)	; EDBUF ; get edit buffer address ; push it
DICTCODE LI	.BYTE .ASCII .WORD EA MOVE.W RTS	4 "DIC" EDBCODE-6 DICT,A0 A0,-(A6)	; DICT ; get dictionary address ; push it
LOAD	.BYTE .ASCII .WORD GETBLOCK RTS	4 "LOA" DICTCODE-6	; LOAD ; system dependent macro
GO	.BYTE .ASCII .BYTE .WORD CLR.B JSR	2 "GO" 0 LOAD-6 FINT CR	; GO ; leave interactive mode ; restart input sequence
SAVE	.BYTE .ASCII .WORD SAVBLOCK RTS	4 "SAV" GO-6	; SAVE ; system dependent macro
TYPE	.BYTE .ASCII .WORD MOVE.W SUBQ.B MOVEA.W MOVE.B	4 "TYP" SAVE-6 (A6)+,D1 #1,D1 (A6)+,A0 (A0)+,D0	; TYPE ; get character count ; ; get buffer address
	TPUT .WORD .WORD RTS	51C9H 0FFF6H	; DO ; send buffer character ; to terminal ; UNTIL exhausted (continued on page 58)

## Turbo Tech Report Speaks Your Language.









#### The newsletter/disk publication for Turbo Pascal®users

Are you a devoted Turbo Pascal programmer, tired of reading about other languages? Are you looking for powerful utilities written in Turbo Pascal that you can use to develop software or incorporate into your programs? Are you interested in improving and expanding your Turbo Pascal programming skills?

Then you deserve a subscription to *Turbo Tech Report*, the bimonthly newsletter/disk publication from the publishers of *Dr. Dobb's Journal* and *Micro/Systems Journal*. Each issue delivers more than 250K of Turbo Pascal source code programs on disk, and 20+ pages of articles, Turbo Pascal software and book reviews, and analysis and commentary. It's the only publication delivering such focused technical articles with code on disk—and it doesn't waste your time with information about other programming languages. Each valuable issue contains:

- *Articles* on topics like speedy 3D graphics, mathematical expression parsers, creating global gotos, memory resident and AI applications and more—all written by Turbo experts.
- *Reviews* of the latest Turbo Pascal software programs from companies like Borland

International, Blaise Computing, Media Cybernetics, Nostradamus, TurboPower Software, and more!

- *News and commentary* detailing the latest products and developments in the Turbo Pascal programming community.
- A disk filled with Turbo Pascal code! You'll get the Turbo Pascal utilities and routines discussed in the newsletter's articles, as well as applications developed by Turbo users from around the world. You'll receive programs that make labels, generate menus, provide faster screen access, transfer files between CP/M and MS-DOS computers, and more!

If you're an expert Turbo Pascal programmer or a novice interested in expanding your Turbo skills, you need a publication that speaks your language: *Turbo Tech Report*. Subscribe today at the special price of just \$99—that's 33% off the regular price of \$150. To order by credit card, call toll-free 1-800-528-6050 ext. 4001 and ask for item 300. Or mail the attached coupon with your payment to *Turbo Tech Report*, 501 Galveston Drive, Redwood City, CA 94063.

Turbo Pascal is a trademark of Borland International Inc.

#### MAKE YOUR PC **SEEM LIKE AN AT!**

**MAKE YOUR AT** SEEM LIKE A DREAM MACHINE!



The Integrated Console UtilityTM

#### **FAST, POWERFUL** ANSI.SYS REPLACEMENT

For the IBM-PC, AT, and clones

New Version 2.00 is MUCH FASTER! Now blink free scrolling on CGA!

Driver

ANSI Console

Fast

Now uses EMS/EEMS for Scroll Recall New Menu Program for Changing Options

#### **GET A BOX FULL OF UTILITIES!** MAKE LIFE EASIER FOR ONLY \$75!

- Speed up your screenwriting 2-6x
   Extend your ANSI.SYS to full VT100
- Add many more escape sequences Scroll lines back onto screen
- Save scrolled lines into a file
- Add zip to your cursor keys
  Free your eyes from scroll blinking
  Easy installation

ANSI

Console

Driver

- Get a 43 line screen w/EGA
  Get a 50 line screen w/CGA
  No more annoying typeahead beep
- Prevent screen phospher burnin
- Control many programs' use of color
   Generate breakpts from keyboard
- Shorten that annoying bell Over 50 other useful options
- "The psychological difference is astonishing'
- —Lotus June 85 pg 8.
- "So many handy functions rolled into one unobtrusive package' -PC-World Feb 86 pg 282.
- "The support provided by the publishers is extraordinary"

  —Capital PC Monitor May 86 pg 25.
- . the best choice for improving your
- console -Capital PC Monitor June 86 pg 26.

460p Manual (w/slip case) & disks \$75.

Satisfaction Guaranteed! Order Yours Today!

HERSEY MICRO CONSULTING Box 8276, Ann Arbor, MI 48107 (313) 994-3259 VISA/MC/Amex

DEALER INQUIRIES INVITED



Circle no. 280 on reader service card.

#### 68000 MINI FORTH

#### Listing Two (Listing continued, text begins on page 22.)

```
. ASCII
                .WORD
                                 TYPE-6
PRINT
               MOVEQ
                                 #13.D0
                                                              send CF
               TPUT
                                 #10,D0
                TPUT
                MOVEA.W
                             A5,A0
                                                              get buffer pointer
               MOVE.L
                                 BLANK, (A0)+
                                 BLANK, (A0) +
BLANK, (A0) +
ZERO, (A0) +
               MOVE. I.
                                                              zero out buffer
               MOVE.L
MOVE.L
                                                              pop number
if negative
               MOVE . W
                                 (A6) +, D2
DLOOP
                BGE
                                                                  negative
                                                               make positive
set negative flag
while quotient not 0 do
                NEG.W
                                 D2
                                 FNEG
                TAS
DLOOP
                ANDI.L
                                 #65535,D2
                                 TSTMINUS
                BEQ
                                                                   clear remainder
DIV
                                                                   divide by base
(do dirty work in DO)
get remainder
               DIVS
                                 BASE, D2
                                 D2, D0
               SWAP
                                 DO
                CMPI.B
                                 #10.DO
                                 PREFIX
#7,D0
                                                                   convert to digit
                ADDQ.B
                                 #48,D0
D0,-(A0)
DLOOP
PREFIX
                                                                   make ASCII prefix
place digit in buffer
                ADDI.B
                MOVE . B
                BRA
                                 #7.FNEG
TSTMINUS BCLR
                                                               test and clear negative flag
                                 #7, FNEG
PRNT
#45, - (A0)
A5, - (A6)
#16, - (A6)
TYPE
               BEO
                                                              put minus sign in buffer
push buffer address
push buffer length
               MOVE . B
PRNT
               MOVE . W
                MOVE.W
                JSR
                                                               type number
               RTS
                .BYTE
                                 2
".s"
                . ASCIT
                                                                   .s
                 BYTE
                                 PRINT-6
                 . WORD
SPRINT
                LEA
                                 UNDRFLW, A1
                                                              get address of bottom
               MOVEA.W
CMPA.W
BOTTOM
                                 A1, A2
DONE
                                                               while above bottom do
                BEO
                MOVE.W
                                 (A2)+,-(A6)
                                                                   push next number
                JSR
                                 PRINT
                                                                   print it
                BRA
                                 BOTTOM
DONE
                BYTE
                                 129
                .WORD
                 WORD
                                 SPRINT-6
                                 (A4)+,D0
#41,D0
CMMNT
                                                              get character from input buffer
until ")"
                CMPI.B
                BNE
                                 CMMNT
                                 4
"QUI"
                .BYTE
                .ASCII
                                                                   QUIT
                                 CMMNT-6
QUIT
                XSTOP
                                                            ; return to monitor
                RTS
LAST
                .BYTE
                                 6
"LOG"
                 .ASCII
                                                                   LOGOFF
                                 QUIT-6
                LEA
                                 BUFPNT, AO
                                                            ; save dictionary pointer
                MOVE.W
RTS
                                 A5, (A0)
                 . WORD
BUFPNT
                                 BUFFER
                .END
```

**End Listing Two** 

#### **Listing Three**

(Listing Three)
("inner shell" words for FLINT)

(The "->" symbol when used in a comment signifies that the) (instruction corresponding to the preceding assembler) (mnemonic will be written in the dictionary at execution time.)

- CONSTANT ( n -- creates a constant) TOKEN HEADER LITERAL CODE 3AFC 4E75 ( rts  $\rightarrow$  ) ;
- CREATE ( -- creates the header and code body for a variable)
  TOKEN HEADER CODE 2AFC 41FA 0006 ( lea 6[a5], a0 -> )

  3AFC 3D08 ( move.w a0,-[a6] ( rts -> ) :
- ( n -- \_ used after CREATE to allocate space for a ) variable or an array +, A5 ); CODE DADE ( ADDA. W [A6]+,A5
- creates a variable) CREATE 2 ALLOT ;
- "fetch" replaces an address with its value) CODE 305E 3D10 ( MOVEA. [A6]+,A0 ) [A0],-[A6] );

```
stores a word length value in the address)
 CODE 305E
               ( MOVEA.W
                                    [A6]+, A0 )
[A6]+, [A0] );
       309E
  : !BYTE
                                    stores a byte length value in the address) [A6]+,A0 )
                                    [A6]+,[A0] ) ;
: HEX
16 BASE ! ;
                            changes the system base to 16)
  DECIMAL ( --
10 BASE ! ;
                            changes the system base to 10)
 : DUP
                  (n -- n n)
  CODE 3D16 ( MOVE. W
                                    [A6],-[A6]);
                                        tests the top value, drops it, and sets CCs)
  CODE 4A5E ( TST.W [A6]+
                                     synonym for "?" used to emphasize the drop)
 \ (n --
CODE 4A5E (TST.W [A6]+
  OVER ( n1 n2 -- n1 n2 n1 )
CODE 3D2E 0002 ( MOVE.W 2[A6],-[A6] );
                  ( n1 n2 -- n1 n2 n1 n2 )
· 2DUP
  OVER OVER :
                                    removes a value from the parameter stack ) and places it on the return stack [A7]+,D1, [A6]+,-[A7]) D1,-[A7]);
                  ( n -- _
: >R
               ( MOVE.L
  CODE 221F
        3F1E
                 MOVE. W
                                    removes a value from the return stack and places it on the parameter stack [A7]+,D1 )
: <R
                  ( _ -- n
              ( MOVE.L
                                   [A7]+,-[A6] )
D1,-[A7] );
        3D1F
                 MOVE. W
  ROT ( nl n2 n3 -- n2 n3 n1 ) >R SWAP <R SWAP ;
: ROT
                  ( n1 n2 -- n1+n2 )
              ( MOVE.W [A6]+,D1 )
( ADD.W D1,[A6] );
  CODE 321E
  ~ (n -- -n)
CODE 4456 (NEG.W[A6]);
                  ( n1 n2 -- n1-n2 )
  * ( n1 n2 -- n1*n2 )
CODE 321E ( MOVE.W [A6]+,D1 )
C3DE ( MULS [A6]+,D1 )
3D01 ( MOVE.W D1,-[A6] )
                                    D1,-[A6] ) ;
                  ( nl n2 -- n1/n2 nl mod n2 )
: /MOD
                 MOVE.W
MOVE.W
                                    [A6]+,D1 )
[A6]+,D2 )
  CODE 321E
        341E
48C2
                 EXT.L D2
        85C1
                 DIVS D1, D2
                 MOVE.W
SWAP D2
        3D02
                                    D2,-[A6] )
        3002
               ( MOVE W
                                    D2. -[A6] ) :
                  ( n1 n2 -- n1/n2 )
  /MOD \ .
                 ( n1 n2 -- n1 mod n2 )
   /MOD SWAP \ ;
                                    returns a true flag if n < 0)
                  ( n -- f
                            ( TST.W [A6]
( BLT 4[PC]
( CLR.W [A6]
  CODE 4A56
        4256
                     ( RTS
                                   #-1,[A6] ) ;
                                   returns a true flag if n > 0)
: < _ 0< ;
                  ( n1 n2 -- f returns a true flag if next < top)
: >
                  ( n1 n2 -- f returns a true flag if next > top)
: CGET
                                        gets a character from the terminal and) places its ASCII value on the stack
                  ( - -- n
  CODE 4EB9 00FE 0008 ( TGET
                                 ( MOVE.W
                                                 DO, -[A6] ) ;
                                      takes an ASCII value from the stack and) sends it to the terminal
        4EB9 00FE 0008 ( TPUT
  CODE 301E
                                                  [A6]+,D0 )
: CLEAR
26 EMIT ;
                  ( --
                           erases the screen)
                             increments CBLOCK: loads and executes
                                         the new block [allows chaining]
                                                                           (continued on page 61)
```

# BRICKLIN'S DEMO BROGRAM

Read what they're saying about this new concept in prototyping and demo-makina:

"A winner right out of the starting gate. Äfter you use DEMO once, you'll wonder how you got along without it.'

PC Magazine, 4/29/86

"Everybody who writes software, either commercially or for in-house applications, should immediately order a copy. Period. No exceptions."

Soft • letter, 4/20/86

"Its low price, superb performance, and range of applications practically guarantee that it will be widely used. Four Floppy Rating (8.0)"

- InfoWorld, 3/31/86

"Apparently has a hit on its hands with ... a development tool for personal computer software that has won rave reviews from early users.'

- Computerworld, 4/7/86

"A gem." - PC Week, 3/18/86

RDER NOW! Thousands of developers are de-

signing better products faster and producing more effective demonstrations using Dan Bricklin's Demo Program. You can, too. Act now!



Massachusetts residents add \$3.75. Outside of the U.S.A. add \$15.00.

Requires 256k IBM PC/compatible, DOS 2.0 or later. Supports Monochrome, Color/graphics, and EGA Adaptors (text mode only).



Dept. D

P.O. Box 373, Newton Highlands, MA 02161

Circle no. 314 on reader service card.

## PAINLESS WINDOWS.

Windows. Data Entry. Menus. Finally, a C programmers' tool that makes them as easy to use as printf(). With Greenleaf DataWindows™, you move in quantum leaps!

#### Snazzy Window Treatments

DataWindows represents an important breakthrough in C programming tools. It sets you free so you can create exciting programs quickly and easily, saving both time and money! Developed to work with the IBM PC, XT, AT, compatibles, and MSDOS or PCDOS, DataWindows is a carefully tooled system of C functions which will jazz up your programs with unprecedented efficiency.

Greenleaf DataWindows is integrated windows, transaction data entry, pop-up, pull-down, and Lotus style menu systems

- Screen Management. You don't have to remember what's on the display or the sequence in which you put it there. DataWindows does the grunt work. There are no restrictions.
- Transaction Data Entry. Data entry windows can have any number of fields with sophisticated options for reading many data types. Calls are made to help, validation, and other functions. Full featured text editing, protected and mandatory fields, dBASE type picture strings, context sensitive help, validation of fields and transactions, redefinable keys, password entry, attribute control, keyboard idle and much more.
- Device Independence. It detects the type of display adapter your computer is using and adjusts to it automatically for CGA, EGA, or monochrome. Logical video attributes are easy to use for color or monochrome.
- Compatibility. Runs with Microsoft Windows and IBM TopView.
- The Greenleaf Tradition of Quality. Reliable products. Professional documentation that gets you up and running quickly and keeps you there. Reference card. Newsletter and Bulletin board.

IBM, Microsoft & dBase, are registered trademarks of International Business Machines, Microsoft Corporation & Asthon-Tate respectively, PCDOS, IBM PC, XT, AT, & TopView are trademarks of IBM; MSDOS and Microsoft Windows are trademarks of Microsoft Corporation.

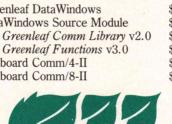


#### Stop Window Shopping

Order Today. Or call toll free for a free demo of the windows library that makes all the others obsolete.

Order any of these high performance tools by calling your dealer or 1-800-523-9830 today. Specify compiler when ordering. Add \$8 for UPS second day air, or \$5 for ground. Texas residents add sales tax. MasterCard, VISA, P.O., check, COD. In stock, shipped next day.

Greenleaf DataWindows \$225 \$225 DataWindows Source Module The Greenleaf Comm Library v2.0 \$185 The Greenleaf Functions v3.0 \$185 Digiboard Comm/4-II \$325 Digiboard Comm/8-II \$535





1411 LeMay Drive, Suite 101 Carrollton, TX 75007

**Call Toll Free** 1-800-523-9830 In Texas and Alaska, call 214-446-8641

Circle no. 97 on reader service card.

#### Window Dressings

- Simple or Complex Windows. Up to 254 powerful overlaid windows simultaneously, all with just one kind of window to remember! Yet any window can be from one character to 32K!
- Easy Window Operations. DataWindows lets you move, zoom, frame, title, change colors, titles, frames, size, location, and make windows visible or invisible at will! Functions set cursor, attributes, and write data to any window or "current window". Word wrap, auto scroll, keyboard functions.
- Write to Any Window Any Time. Windows may be visible, overlaid, or invisible, and you can write to them anyway. What you write will be seen when the windows become visible.
- DataWindows is fast! It writes directly to video memory (in some modes).
- **Easy to save!** Any window, complete with attributes, can be saved on disk quickly and efficiently.
- Source code available. No royalties.

#### Also from Greenleaf:

#### The Greenleaf Functions v3.0

The most complete, mature C language function library for the IBM PC, XT, AT and close compatibles. Includes over 225 functions — DOS, disk, video, color text and graphics, string, time/date, keyboard, disk status and Ctrl-Break functions plus many more.

The Greenleaf Comm Library Our 2.0 version is the hottest communications facility of its kind. Over 120 functions — ring buffered, interrupt driven asynchronous communications for up to 16 ports simultaneously with XMODEM, XON/XOFF, many many sophisticated features.

We support all popular C compilers for MSDOS/PCDOS: Microsoft, Lattice, Computer Innovations, Aztec, DeSmet, and others.

#### 68000 MINI FORTH

Listing Three (Listing continued, text begins on page 22.)

```
CBLOCK @ 2 +
                                CBLOCK ! LOAD
  ?> ( -- )
CODE 3AFC 4A5E ( tst.w
                                           [a6]+ ->);
: BRA> ( -- )
CODE 3AFC 6000 ( bra
  BEQ> ( -- )
CODE 3AFC 6700 ( beq
: BNE> ( -- )
CODE 3AFC 6600 ( bne
                                              pushes the contents of the dictionary pointer) A5,-[A6] );
   SPLIT ( --
?> BEQ> MARK 2 ALLOT ;
  JOIN ( -- )
DUP MARK SWAP - SWAP ! ;
 IF (: SPLIT ; IMMEDIATE -
   THEN (:
JOIN ; IMMEDIATE
: ELSE ( : a1 -- a2 x
BRA> MARK 2 ALLOT SWAP JOIN ; IMMEDIATE
  DO (: MARK; IMMEDIATE
: UNTIL (: a -- x
?> BNE> MARK - , ; IMMEDIATE
: WHILE (: --- a SPLIT ; IMMEDIATE
  LOOP (: a1 a2 -- x -
BRA> SWAP MARK - , JOIN ; IMMEDIATE
  ' ( \, pushes the address of the token which follows ') TOKEN DICT \mbox{\it @} SEARCH IF ELSE WHAZZAT THEN ;
   FORGET ( -- erases the dictionary entries for the token following)

( FORGET as well as all tokens which succeed it in the )
( dictionary

*DUP 2 - @ DICT ! 6 - CODE 3A5E ( MOVEA.W (A6)+,A5 ) ;
```

**End Listings** 

## A special offer to our readers ...

# A SPECIAL SOFTSTRIP® READER OFFER FROM:

DR. DOBB'S JOURNAL OF SOFTWARE TOOLS

The price: \$199.95 complete

## A SPECIAL To order: Call toll free 1-800-528-6050 ext. 4001.

Refer to product item #400. Or send this coupon with payment to Dr. Dobb's Journal of Software Tools, 501 Galveston Dr., Redwood City, CA 94063. Please be sure to specify software format (MS/DOS, Macintosh, Apple II). CA residents add appropriate sales tax.

Please specify format:

☐ MS-DOS	☐ Macintosh	☐ Apple II

Payment enclosed (check or money order)

VISA	VISA	MasterCard

Card No. Exp. Date

Signature

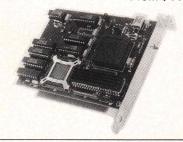
Name

Address

City State Zip

#### MICROWAY MEANS 8087 PERFORMANCE

#### FastCACHE-286™



#### LOTUS/INTEL EMS SPECIFICATION BOARDS

MegaPage™ The only EMS board which comes populated with two megabytes of cool-running, low power drain CMOS RAM installed. Includes RAM disk, print spooler, disk cache and EMS drivers. For the IBM PC, XT and compatibles...\$549

MegaPage with ØK.....\$149

MegaPage AT/ECC™ EMS card for the PC AT and compatibles includes Error Correction Circuitry. With ECC, 11 RAM chips cover 256K so the user never encounters RAM errors Sold populated with 1 megabyte CMOS . . . \$699 or with 3 megabytes CMOS cool running low power drain RAM . . . \$1295. Optional serial/parallel daughterboard. . . . . \$95

#### NUMBER SMASHER/ECM™



PC Magazine "Editor's Choice"

#### DATA ACQUISITION and REAL TIME TOOLS

DAL™ – "Data Acquisition Language."

Unkelscope™ – A real time data acquisition, control and process software pkg.

87 FFT and 87 FFT-2

**TransView** Menu driven FFT Spectrum/ transfer analyzer.....\$250

INTEL COMPILERS Available for RTOS FORTRAN-86, PASCAL-86, PL/M-86.

#### 287 Turbo™-10/12



PC Magazine "Editor's Choice"

#### 8087 UPGRADES

All MicroWay 8087s include a one year warranty, complete MicroWay Test Program and installation instructions.

8087-2 8 MHz ...... \$149 For Wang, AT&T, DeskPro, NEC, Leading Edge 80287-3 5 MHz ..... \$179

For the IBM PC AT and 286 compatibles

**80287-6 6 MHz**.....**\$229** For 8 MHz AT compatibles

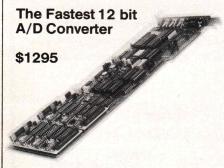
**80287-8 8 MHz** ..... \$259 For the 8 MHz 80286 accelerator cards

80287-10 10 MHz.....\$395

For the Compaq 386

Call for prices on V20, V30, 64K, 128K and 256K RAM

#### A2D-160™



#### 8087 SOFTWARE

IBM BASIC COMPILER	\$465
MICROSOFT QUICK BASIC	\$79
87BASIC COMPILER PATCH	\$150
IBM MACRO ASSEMBLER	\$155
MS MACRO ASSEMBLER	\$99
87MACRO/DEBUG	
MICROSOFT FORTRAN	
RM FORTRAN	. \$399
LAHEY FORTRAN F77L	. \$477
MS or LATTICE C	CALL
STSC APL★PLUS/PC	
STSC STATGRAPHICS	
SPSS/PC+	\$675
87SFL Scientific Functions	\$250
PHOENIX PRODUCTS	
FASTBREAK for 1-2-3 V.1A	\$79
HOTLINK for 1-2-3 V.1A	

#### 287 TURBO-PLUS™ Speeds up your AT

Adjustable 80286 Clock 6-12 MHz 10 MHz 80287 Clock Plus Full Hardware Reset.......\$149 Optional 80286-10



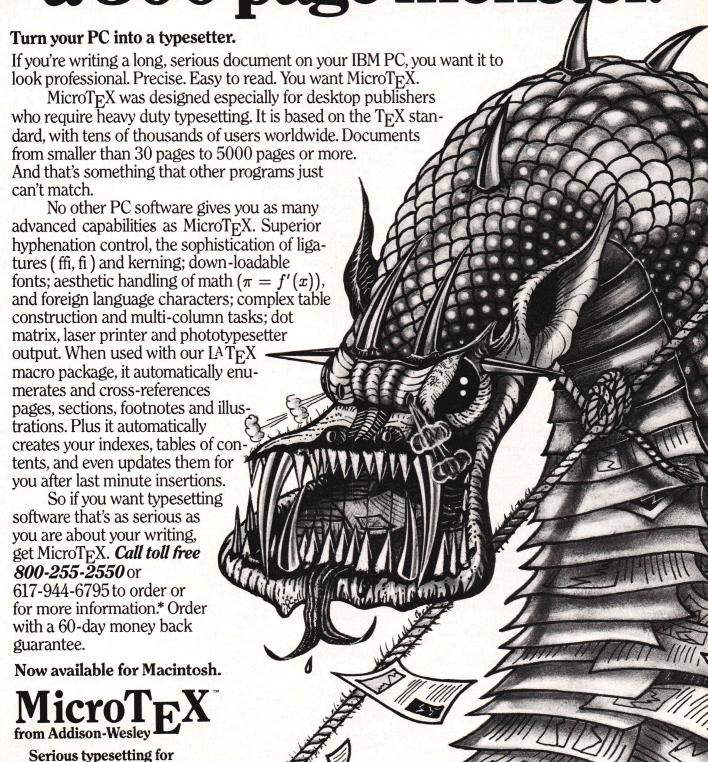
With 80287 10 MHz.....

#### CALL (617) 746-7341 FOR OUR COMPLETE CATALOG

Micro P.O. Box 79 Kingston, Mass. 02364 USA (617) 746-7341

The World Leader in 8087 Support!

MicroWay Europe 32 High Street Kingston-Upon-Thames Surrey England KT1 1HL Telephone: 01-541-5466 How to tackle a 300 page monster.



serious desktop publishers.
\*Dealers, call our Dealer Hot Line: 800-447-2226

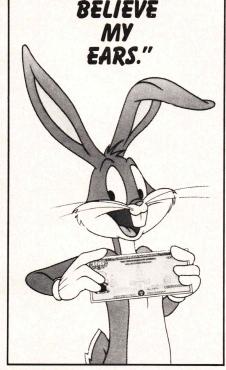
(In MA, 800-446-3399), ext. 2643. Circle **no. 92** on reader service card.

#### MAC AND AMIGA

#### Listing One (Text begins on page 40.)

```
Listing One
                     Include
                                MacTraps.D
2
                     Include
                                ToolEqu.D
3
                     Include
                                SysEqu.D
4
                     Include
                                QuickEqu.D
                      InitGraf
                     PEA
                                                                ; initialize managers
6
                      InitFonts
8
                     MOVE.L
                               #$0000FFFF.D0
                      FlushEvents
                     __InitWindows
10
11
                      InitMenus
12
                     CLR.L
                                - (SP)
                     _InitDialogs
13
14
                      InitCursor
                     CLR
                                - (SP)
16
                                'DrD.Rsrc'
                     PEA
17
                      OpenResFile
                                                                ; open resource file
                     MOVE
18
                                (SP) + D0
                                                                ; discard unused result
                                - Set up menus
19
                     CLR.L
                                - (SP)
                                                                ; space for handle
20
                     MOVE
                                #1,-(SP)
                                                                ;menu ID
21
                      GetRMenu
                                                                ;get Apple menu template
22
                     MOVE.L
                                (SP) +, AppleHandle (A5)
                                                                ; retrieve & store handle
                                AppleHandle (A5), -(SP) #'DRVR', -(SP)
23
                     MOVE.L
                                                                ;put handle back on stack
;res type for desk accs
;get desk accessories
                     MOVE.L
25
                     AddResMenu
26
                     MOVE.L
                                AppleHandle (A5), - (SP)
27
                                - (SP)
                                                                ; put menu after all others
28
                     _InsertMenu
                                                                ; put menu in menu list
                     CLR.L
                                - (SP)
                                                                repeat for other menus
30
                     MOVE
                                #2,-(SP)
                      GetRMenu
31
32
                     MOVE.L
                                (SP)+,FileHandle (A5)
33
                     MOVE.L
                                FileHandle (A5), - (SP)
34
                     CLR
                                - (SP)
35
                      InsertMenu
36
                     CLR.L
                                - (SP)
37
                     MOVE
                                #3,-(SP)
38
                      GetRMenu
39
                     MOVE.L
                                (SP)+, EditHandle (A5)
                     MOVE.L
                                EditHandle (A5), - (SP)
41
42
                     _InsertMenu
43
                      DrawMenuBar
                      - Open the window with a text edit record --
                     CLR.L
                                -(SP)
#1,-(SP)
                                                                ; space for window pointer
45
                     MOVE
                                                                ;window ID
46
                     PEA
                                WindowStorage (A5)
                                                                ;window storage
                     MOVE.L
                                #-1,-(SP)
                                                                ; put window in front
48
                      GetNewWindow
49
                     MOVE.L
                                (SP)+,WindowPtr(A5)
                     MOVE.L
                                WindowPtr(A5),-(SP)
51
                      SetPort
                                                                ; makes window current grafport
52
                     CLR.L
                                - (SP)
                                                                ;place for text handle
                     PEA
53
                                Dest Rect
                                                                ;destination rectangle
54
                     PEA
                                ViewRect
                                                                ; view rectangle
55
                      TENew
                                                                ;establish text edit record
56
                     MOVE.L
                                (SP) + . TextHandle (A5)
                                                                ;get handle
57
                     MOVE.L
                                TextHandle (A5), - (SP)
58
                     _TEActivate
                                                                ; make text edit record active
:
                     -- Event loop begins here -
          Event
59
                      SystemTask
                                                                ; update desk accessories
                     MOVE.L
60
                                TextHandle (A5), - (SP)
61
                     TEIdle
                                                                ; make cursor blink
62
                     CT.D
                                                                ; space for boolean result
                                #-1,-(SP)
EventRecord(A5)
63
                     MOVE
                                                                ; mask to select all events
64
                     PEA
                                                                ; pointer to event record
65
                     GetNextEvent
                                                                ;get event from queue
66
                                (SP)+,D0
                     MOVE
                                                                ; retrieve boolean result
67
                     BEQ
                                                                :no event
68
                     MOVE
                                EventRecord (A5), D0
                                                                ; get event type
                                                                                                                     (continued on page 66)
```

"WHEN I
HEARD ABOUT
U.S. SAVINGS BONDS'
COMPETITIVE
RATE...
I COULDN'T



©WARNER BROS. INC. 1986 Used With Permission.

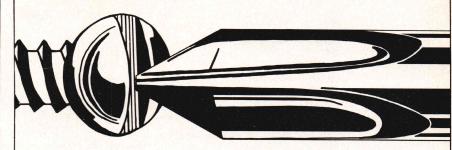
Everyone's amazed when they hear what U.S. Savings Bonds have to offer. Savings Bonds earn market-based interest rates—just like money market accounts—so you're guaranteed a competitive return, no matter what happens to interest rates. All you have to do is hold your Bonds for five years. What's more, the interest you earn is completely free from state and local income taxes. And federal taxes can be deferred. For more information, call 1-800-US-BONDS. U.S. Savings Bonds— They'll make your ears perk up.

U.S. SAVINGS BONDS

Bonds held less than five years earn a lower rate.

A public service of this publication.

## ISN'T IT A PITY...



## **Everything Isn't As Accommodating As**

c-tree / r-tr

FILE HANDLER

REPORT GENERATOR

#### Performance and Portability

For all the time you devote to developing your new programs, doesn't it make sense to insure they perform like lightning and can be ported with ease?

#### c-tree: Multi-Key ISAM Functions For Single User, Network, & Multi Tasking Systems

Based on the most advanced B+ Tree routines available today, **c-tree** gives you unmatched keyed file accessing performance and complete C Source Code. Thousands of professional C programmers are already enjoying **c-tree**'s royalty-free benefits, outstanding performance, and unparalleled portability.

Only FairCom provides single and multi-user capabilities in one source code package, including locking routines for Unix, Xenix, and DOS 3.1., for one low price! In addition, **c-tree** supports fixed and variable record length data files; fixed and variable length key values with key compression; multiple indices in a single index file; and automatic sharing of file descriptors.

#### r-tree: Multi-File Report Generator

r-tree builds on the power of c-tree to provide sophisticated, multi-line reports. Information spanning multiple files may be used for display purposes or to direct record selection. You can develop new reports or change existing reports without programming or recompiling and can use any text editor to

create or modify **r-tree** report scripts including the complete report layout. At your option, end users may even modify the report scripts you provide.

#### Unlimited Virtual Fields; Automatic File Traversal

r-tree report scripts can define any number of virtual fields based on complex computational expressions involving application defined data objects and other virtual fields. In addition, r-tree automatically computes values based on the MAX, MIN, SUM, FRQ, or AVG of values spread over multiple records. r-tree even lets you nest these computational functions, causing files from different logical levels to be automatically traversed.

Unlike other report generators, **r-tree** allows you to distribute executable code capable of producing new reports or changing existing reports without royally payments, provided the code is tied to an application. Your complete source code also includes the report script interpreter and compiler.

#### **How To Order**

Put FairCom leadership in programmers utilities to work for you. Order **c-tree** today for \$395 or **r-tree** for \$295. (When ordered together, **r-tree** is only \$255). For VISA, MasterCard and C.O.D. orders, call 314/445-6833. For **c-tree** benchmark comparisons, write FairCom, 2606 Johnson Drive, Columbia, MO 65203.



Complete C Source Code & No Royalties!

Xenix is a registered trademark of Microsoft Corp. Unix is a registered trademark of AT&T

Circle no. 93 on reader service card

#### MAC AND AMIGA

#### Listing One (Listing continued, text begins on page 40.)

69 70	CMP BEQ	#mButDwnEvt,D0 MouseEvent	;mouse event?
71 72	CMP BEQ	#keyDwnEvt,D0 KeyEvent	;key pressed?
73 74	CMP BEQ	#upDatEvt,D0 Update	;refresh?
75	BRA	Event	;no desired event posted
;	H	andle key down events	
76	eyEvent MOVE	EventRecord+evtMeta(A5),D0	
77 78	BTST BNE	#cmdKey,D0 KeyboardEquivalent	;command key pressed?
79 80 81	MOVE MOVE.L TEKey	EventRecord+evtMessage+2(A5) TextHandle(A5),-(SP)	,-(SP) ; character pressed ;insert character
82	BRA	Event	, indicate state of the state o
Ke	eyboardEquivalent		
83 84 85	CLR.L MOVE _MenuKey	- (SP) EventRecord+evtMessage+2 (A5)	<pre>;place for menu ID &amp; item nubmer ,-(SP) ;character ;which menu?</pre>
86	BRA	Selections	
;		Update the text window	
87 Ur	odate MOVE.L	WindowPtr(A5),-(SP)	
88	_BeginUp		
89 90	MOVE.L _SetPort	WindowPtr(A5),-(SP)	
91 92 93	PEA MOVE.L _TEUpdate	ViewRect TextHandle(A5),-(SP)	
94 95	MOVE.L _EndUpdat	WindowPtr(A5),-(SP) te	
96	BRA	Event	
;		Handle mouse down events	
97 Mc	ouseEvent CLR	-(SP)	;space for "what" result
98 99	MOVE.L PEA	EventRecord+evtMouse (A5), - (S WhichWindowPtr(A5); window a	SP) ;place where event occurred
100	_FindWin	dow	;get exact location of event
101	MOVE	(SP)+,D0	;recover result
102 103	CMP BEQ	#inMenuBar,D0 MenuBar	;in menu bar?
104 105	CMP BEQ	#inSysWindow,D0 SysEvent	;in desk accessory?
106 107	CMP BEQ	#inContent,D0 ApplWindow	;in text edit area?
108 109	CMP BEQ	#inGoAway,D0 GoAwayBox	;in close box?
110	BRA	Event	; not an event this program handles
;	Handle ever	nts in system windows	
111 Sy	vsEvent PEA	EventRecord (A5)	
112 113	MOVE.L _SystemC	WhichWindowPtr(A5),-(SP)	;window posting event ;let system handle it
114	BRA	Event	
;	Handle	events in content area of wir	ndow
115 116	pplWindow PEA _GlobalT	EventRecord+evtMouse (A5) oLocal	;event location ;convert coordinates
117	MOVE.L	EventRecord+evtMouse (A5), - (S	SP) ; coordinates now local
118 119	MOVE BTST.L	EventRecord+evtMeta(A5),D0 #shiftKey,D0	;extended selection?
120 121	SNE MOVE.B	D0 D0, - (SP)	
122	MOVE.L	TextHandle (A5), - (SP)	; extend or not extend
124	_TEClick	The state of the s	;set selection range
	BRA	Event	

(continued on page 68)

## C DYNAMO

FREE! UNIX<sup>†</sup> WORKALIKE FREE!

(See Limited Time Special Offer Below)

#### NEW **DYNAMO SCREEN** PAINTER AND FORMS CREATOR

DOES IT ALL RIGHT FROM YOUR KEYBOARD **AUTOMATIC CODE GENERATION** 

Data or Help Screens & Windows Data Entry Screens & Windows Menus

FAST, FLEXIBLE, EASY

Save Man-Months of Programming Full Control of Screen Attributes Monochrome or Color by Form, Screen, Window or Item

**FAST EASY MENU GENERATION** 

1-2-3 Like, Many Others Full Read/Write Security by Item

Requires Dynamo Data Entry Screen painter & manual

#### **B-TREE LIBRARY** & ISAM DRIVER

**POWERFUL DATA MANAGER** 

Fixed/Variable length records Fast! Easy To Use! 16.7 Million Records/File 16.7 Million Keys/File

Full source. No royalties. \$129.95

MAKE Utility (Snake) \$59.95

#### SUPERFONTS FOR C

Super Size Characters Monochrome adapter Color/graphics adapter 8 Font Libraries

Font and Function Library \$49.95

#### C-TERP

#### INTERPRETER FOR C

No Compromise, Full K&R Built In Screen Editor Fast, Fast Compile/Link Use External Libraries! Symbolic Debugging Single Stepping Rave Reviews!

2 disks and manual

\$299.95

#### COMBINE AND SAVE!

PAINT TEXT, HELP, MENU & DATA ENTRY SCREENS & WINDOWS FROM KEYBOARD • COMPLETE INPUT FORMATTING

FULL VALIDATION • UNLIMITED WINDOWS • AUTOMATIC
WINDOW OVERLAY & RESTORE • FULL ATTRIBUTE CONTROL & HIGHLIGHTING • 1-2-3 LIKE MENUS • DRAW BOXES
& BORDERS • GENERATE "C" CODE WITH TOUCH OF A KEY
• LEFT/RIGHT/CENTER JUSTIFICATION • SPECIFY ANY FILL

CHARACTER & CURRENCY SYMBOL • U.S./EUROPEAN NUMBER STYLES • MORE DATA TYPES • ONE CALL WILL OPERATE ENTIRE SYSTEM • SUPPORTS IBM PC/XT/AT & COMPATIBLES • SUPPORTS MICROSOFT, COMPILER LAT-

TICE, AZTEC, CI86, ANY FULL K & R OR ANSII COMPILER

SCREEN PAINTER + DATA ENTRY BOTH for only ...........\$179.95

C LIBRARY plus POWER WINDOWS 

C BUSINESS LIBRARY INCLUDES C FUNCTION LIBRARY, POWER WINDOWS, SUPERFONTS FOR C, B-TREE LIBRARY, ISAM

ALL for .....\$299.95 (A \$440 VALUE + FREE UNIX WORKALIKE)

C TOTAL LIBRARY INCLUDES C FUNCTION LIBRARY POWER WINDOWS, SUPERFONTS FOR C, B-TREE LIBRARY, ISAM and C-TERP C interpreter

C DYNAMO LIBRARY #1 SCREEN PAINTER, DYNAMO DATA ENTRY, POWER WINDOWS, C FUNCTION LIBRARY, SUPERFONTS for C ALL for.....\$249.95

C DYNAMO LIBRARY #2
ALL OF C DYNAMO LIBRARY #1 PLUS B-TREE & ISAM . . . . . . . \$34 (A \$700 VALUE + FREE UNIX WORKALIKE) .\$349.95

C DYNAMO LIBRARY #3
ALL OF C DYNAMO LIBRARY #2
PLUS C-TERP ..........\$549.95
(A \$1000 VALUE + FREE UNIX WORKALIKE)

\*\*FREE PC UNIX, PCVMS OR O/S TOOL BOX WITH ANY PURCHASE OVER \$295.00 LIMIT ONE PER CUSTOMER

Expires 1/31/87

#### SPECIAL OFFER **OPERATING SYSTEMS**

Multi-User Multi-Tasking Networking, Full Source

LIST FRFF\* PCNX™ UNIX WORKALIKE \$99 PCVMS™ 99 FREE O/S TOOL BOX™ (Build your FREE\* own operating system)

™WENDIN SOFTWARE \*with purchase of any Entelekon combination library priced at \$295 or more.

Limit 1 Special Offer item per customer.

Expires 1/31/87

THE C DYNAMO FAMILY...C TOOLS THAT WORK TOGETHER

Enteleko

12118 Kimberley, Houston, TX 77024

713-468-4412

VISA-MASTERCARD-CHECK-COD

#### NEW **DYNAMO DATA ENTRY**

#### UNIQUE, POWERFUL, NECESSARY

Full Validation of Each Keystroke Range Checking Data Security to Item Level Scrollable Data Entry Forms with Fixed & Scrollable Parts Allows Forms Larger Than Screen

#### **Over 34 Item Types**

Powerful "Picture" Capability Unique: Mix Text, Data Entry Fields With Menu Items

Full source code. No Royalties Code plus manual

\$129.95

#### **POWER WINDOWS**

#### MOST POWERFUL YET POP-UP WINDOWS FOR

Menus/Overlays Help Screens Messages/Alarms

ZAP ON/OFF SCREEN FILE-WINDOW MANAGEMENT COMPLETE CONTROL OF:

> Cursor Attributes **Borders**

#### **AUTOMATIC**

Horizontal & Vertical Scrolling Word Wrap Line Insertion

The most powerful, flexible and easy to use windowing package available! Many types of menus. Highlighting. Move data between files, keyboard, program and windows. Status lines. Change size/location/overlapping. Move/add/delete/cascade windows. Full source code. No royalties. 3 disks \$129.95

#### **C FUNCTION LIBRARY**

#### **BEST YOU CAN GET** 325 FUNCTIONS SUPERB DOCUMENTATION

Most complete screen handling plus graphicscursrsor/keyboard/data entry, 72 string functions with word wrap; status andcontrol; utility/DOS BIOS/ time/date functions; printer control & more. Special functions. Full source code. No royalties. 4 disks \$129.95

**+TM AT&T** 

PC-DOS program lets your PC Read/Write/Format over 275 formats



\$79.95 + \$5.00 S/H Sales Tax if CA.

Upgrades available from previous versions

To Order Contact:

1454 Sixth Street, Berkeley, CA 94710

(415) 525-3113

Circle no. 225 on reader service card.

### Quelo® 68000 Software Development Tools

Quelo Assembler Packages are Motorola compatible. Each package includes a macro assembler, linker/locator, object librarian, utilities for producing ROMable code, extensive indexed typeset manuals and produces S-records, Intel hex, extended TEK hex, UNIX COFF and symbol cross references. Portable source written in "C" is available. It has been ported to a variety of mainframes and minis including VAX.

68020 Assembler Package
For CP/M-86, -68K and MS/PC-DOS ....... \$ 750

68000/68010 Assembler Package For CP/M-80, -86, -68K and MS/PC-DOS . . . . \$ 595

68000 "C" Cross Compiler

For MS/PC-DOS by Lattice, Inc.
With Quelo 68000/68010 Assembler Package \$1095
With Quelo 68020 Assembler Package . . . . . \$1250

Call Patrick Adams today:

Quelo, Inc.
2464 33rd W. Suite #173
Seattle, WA USA 98199
Phone 206/285-2528
COD, Visa, MasterCard
Telex 910-333-8171

Trademarks: CP/M, Digital Research; MS, Microsoft Corporation; Quelo,

Circle no. 355 on reader service card.

#### Dr. Dobb's Journal

#### Subscription Problems? No Problem!



Give us a call and we'll straighten it out. Today.

Outside California
CALL TOLL FREE: 800-321-3333

Inside California CALL: 619-485-6535 or 6536

#### MAC AND AMIGA

#### Listing One (Listing continued, text begins on page 40.)

		Handle eve	ents in menu bar	
,	MenuBar			
125 126		CLR.L MOVE.L	- (SP) EventRecord+evtMous	;space for menu ID & item e(A5),-(SP) ;place where event occurred
127		_MenuSele		;get menu ID & item
	Selection			
128 129		MOVE.L MOVE	(SP)+,D7 ;recover : D7,D6	result ;D6 now has menu item
130			D7	;low-order word has menu ID
131 132		CLR _HiLIteMe		;selects all menus ;remove highlighting
133 134		CMP BEQ	#1,D7 *AppleMenu	;apple menu?
135 136		CMP BEQ	#2,D7 FileMenu	;file menu?
137 138		CMP BEQ	#3,D7 EditMenu	;edit menu?
139		BRA	Event	
	Наг	ndle desk	accessories	来的是EEEEC 142.75米 1150 EEEE 1152 不 1915 115
140	AppleMenu			
140 141			AppleHandle (A5), - (S D6, - (SP) ; menu iter	
142 143		PEA _GetItem	DeskAccName (A5)	;space for desk accessory name
144		CLR	-(SP)	;space for reference number
145 146		PEA OpenDesk	Acc ,	desk accessory name; open the desk accessory
147		MOVE	(SP)+,D0	;discard result
148		BRA	Event	
;		Handle	editing	
149	EditMenu	SUBQ	#1,D6	;adjust item selected for SysEdit
150 151		CLR MOVE	-(SP) D6,-(SP)	;space for result ;adjusted item number
152		_SysEdit	50, (51)	, adjusted real number
153 154		MOVE BNE	(SP)+,D0 Event	;get result ;system handled edit
155		ADDQ		restore item nuamber
156 157		CMP BNE	#3,D6 EditMenu2	; cut?
158 159		MOVE.L TECut	TextHandle (A5), - (SP	)
160		BRA	Event	
	EditMenu2			
161 162		CMP		; copy?
163		MOVE.L	EditMenu3 TextHandle(A5),-(SP	)
164 165		TECopy BRA	Event	
	EditMonu2			
166	EditMenu3	CMP		;paste?
167 168		BNE MOVE.L	EditMenu4 TextHandle(A5),-(SP	
169		_TEPaste	reactionate (A5), - (St	
170		BRA	Event	
171	EditMenu4	CMP	#6,D6	·clory?
172		BNE	Event	;clear?
173 174		MOVE.L TEDelete	TextHandle (A5), - (SP	
175		BRA	Event	
;		Mandle file	e menu	
176	FileMenu	CMP	#2,D6	;close selected?
177		BEQ	CloseAndQuit #4,D6	
179				;quit selected
;!!!!!all other file menu options are not implemented in this program!!!!				
180		BRA	Event	
;		Close the	window and quit	
	GoAwayBox			

in all	The state of the s		
181 182 183 184	MOVE.L	WhichWind	;space for boolean result adowPtr(A5),-(SP) ;window pointer cord+evtMouse(A5),-(SP) ;point of event ;monitor GoAway box
185 186			;get result;don't close
187 188	CloseAndQuit MOVE.L _TEDispo		ile(A5),-(SP) ;close text edit record
189 190		WindowPt: ndow	r(A5),-(SP);close the window
191	RTS		;return to Finder
192	AppleHandle FileHandle EditHandle	DS.L DS.L	1 1
195 196	WindowPtr DS.L WindowStorage		WindowSize
197 198 199	TextHandle ViewRect DC DestRect DC		,490
200 201 202	EventRecord WhichWindowPtr DeskAccName	DS.B DS.L DS	16 1 16

**End Listing One** 

#### **Listing Two**

Listi	g Two	
1	DrD.Rsrc	
2 3 4	TYPE MENU ;menu templates follow ,1 ;resource ID \14 ;will create Apple icon for title	
5 6 7 8 9 10 11 12 13 14	,2 ;resource ID File ;menu title New/N ;all the rest are menu items Open/O Close/W Save As Save/S Page Setup Print/P Quit/Q	
15 16 17 18 19 20 21 22	,3 ;resource ID Edit ;menu title Undo/Z (- ;straight line - disabled Cut/X Copy/C Paste/V Clear	
23 24 25 26 27 28 29	TYPE WIND ; window templates follow ,1 ; resource ID Dr. Dobb's Journal 50 10 310 502 ; initial coordinates Visible GoAway ; make window visible, draw GoAway box 0 ; window type (standard document window) 0 ; optional reference number  End Listing Two	

#### **Listing Three**

Listin	g Three			
1 2 3		include include include	"exec/ty "exec/ex "intuiti	
4 5 6	callsys	macro CALLIB endm	_LVO\1	; calls a system routine
7 8 9	xlib	macro xref endm	_LVO\1	;for library routines (continued on page 70

#### DeSmet C SUPPORTS ROMABLE CODE

#### \*\$109 DeSmet C

Data & Stack in RAM Code executes out of ROM Full K&R + V7 extensions Inline asm Assembler, linker, librarian Full screen editor (SEEtm), 8087 & S/W floating point

#### DeSmet C with \$159 D88 Debugger

Set Breakpoints by line number or function name Examine Global/Local variables by name Show C source while debugging Both D88 and User displays

#### Large Memory \$209 DeSmet C w/ D88

32-Bit Pointers (Full Megabyte Addressing) Fast Access of Static Data Includes the standard compiler features mentioned above.

\*Both D88 & Large Memory available as options at \$50 each. We also have a C compiler for the Mac. \$150. Call for details.

- plus -

Graphics \$35 BASICA-like (+src)

Hacker Source for start-up,

RAM.COM & more Make

Full UNIX-level

\$35 Tools (+source)

XARRAY \$39

Large Arrays (+source)

#### **CWARE** CORPORATION

P.O. Box C SUNNYVALE, CA 94087 USA (408) 720-9696 TELEX: 358185 Street Address: 505 W. Olive, #767 VISA, MC & AMEX accepted

\$25

\$50

#### Software Developers: FAR EAST BUSINESS OPPORTUNITY

Kanematsu-Gosho, a prominent Japanese trading company, in conjuction with Tokai Create, one of Japan's largest software marketing firms are soliciting submissions of business related applications software for consideration for export to the Japanese market.

The Japanese PC market as presently approximately 2.5 million units with an anticipated annual growth rate of 25% over the next five years. This offer presents a unique opportunity for U.S. software developers to enter this burgeoning market which has previously been difficult to break into.

Your Japanese partners will be taking care of all the translation, marketing and sales functions in this most interesting market. Successful development companies will be rewarded with a substantial revenue stream for the term of the contract with Japan.

Submitted products will be subjected to a series of evaluations. The initial U.S. based screening will be conducted by **CSSL**, **Inc.** the U.S. representative of the Japanese principals. Successful products will be forwarded to the U.S. offices of Kanematsu-Gosho for further testing and evaluation. Final evaluation will be completed by the parent companies in Tokyo.

Please submit full working versions of your applications software no later than **February 28**, **1987** to:

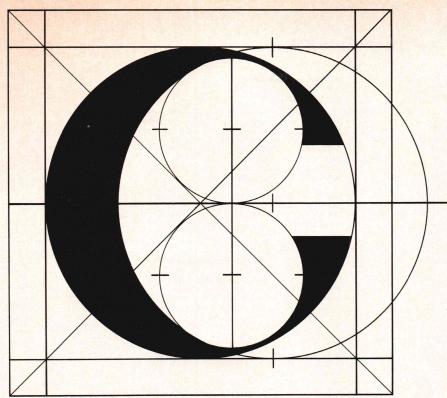
CSSL, INC.
909 Electric Avenue
Seal Beach , CA 90740
Attn: Frank Westall, chairman
Telephone inquiries: 213-493-2471

ALL SUBMISSIONS WILL BE HELD IN THE STRICTEST CONFIDENCE.

#### MAC AND AMIGA

#### Listing Three (Listing continued, text begins on page 40.)

10	passtext	macro			
11		lea	\1,A0	pointer to text	
12		lea	\2,A1	ptr to Intuition text struct	ure
13		jsr	SetText	initializes text structure	
14		endm			
15	passitem	macro			
16	passicen	lea	\1,A0	pointer to menu item structu	ire
17		move.1	\2,A1	pointer to next menu item in	
18		lea	\3,A2	pointer to text structure	
19		move.b	\4,D0	keyboard equivalent	
20 21		move	\5,D1 SetItem	offset from top of menu item initializes menu item struct	
22		endm	Secreen	, initializes menu item struct	uie
23		xlib		l ;external refs for all sys	
24		xlib		routines that the program wi	ill call
25 26		xlib xlib	FreeSigna AddPort		
27		xlib	NewList		
28		xlib	FindTask		
29		xlib	OpenLibra	У	
30		xlib	OpenWindo		
31 32		xlib xlib	SetMenuSt		
33		xlib	OpenDevic		
34		xlib	SendIO		
35		xlib	Wait		
36		xlib	GetMsg		
37		xlib	ReplyMsg		
38 39		xlib xlib	CloseWind		
40		xlib	CloseScre		
41		xref		ise ; exec's base is fixed	
42 43	FrontPen BackPen		0	; for rendering window and te	vt .
43	backreii	equ	1	, for rendering window and ce.	
;		Open Intu	ition lib	ary	
44		move.1	_AbsExect		
45		lea	IntName,	; name of library to	open
46 47		move.l	#0,D0 OpenLibra		
48		callsys	Continue	·Y	
49		rts	Concinae		
				; unsuccessful opening ends p	rogram
				tunsuccessful opening ends p	rogram
50 51	Continue	clr.l	IntBase		
50 51	Continue		IntBase DO,IntBas		
51 ;	Continue	clr.l move.l	DO, IntBa	e ;save base of Intu	ition library
51 ; 52	Continue	clr.l move.l Create	DO, IntBase a custom TheScree	screen	ition library data structure
51 ; 52 53	Continue	clr.l move.l Create lea move	DO, IntBase a custom TheScree #0, ns_Le	; save base of Intu: screenA0 ; pointer to screen Edge (A0) ; coordinates of scr	ition library data structure
51 ; 52	Continue	clr.l move.l Create	DO, IntBase a custom TheScree #0, ns_Le #0, ns_To	; save base of Intuination ; screen	ition library data structure
51 ;52 53 54	Continue	clr.l move.l Create lea move move	DO, IntBase  a custom  The Scree  #0, ns_Le  #0, ns_To  #320, ns_1	; save base of Intuination ; screen	ition library data structure
51 52 53 54 55 56 57	Continue	clr.l move.l Create lea move move move	D0,IntBase a custom TheScree #0,ns_Le #0,ns_To #320,ns_ #200,ns_ #2,ns_De	; save base of Intuination ; screen	ition library data structure
51 52 53 54 55 56 57 58	Continue	clr.l move.l Create lea move move move move move move move	D0,IntBase a custom TheScree #0,ns_te #0,ns_To #320,ns_ #200,ns_ #0,ns_De	; save base of Intuination (A); pointer to screen dege (A0); coordinates of screen dege (A0) dath (A0); color for details	ation library  data structure een
51 ;	Continue	clr.1 move.1 Create lea move move move move move move move.b move.b	D0,IntBase a custom TheScree #0,ns_Le #0,ns_To #320,ns_ #200,ns_ #2,ns_De #0,ns_De #1,ns_Bl	; save base of Intu: screen	ation library  data structure een
51 ; 52 53 54 55 56 57 58 59 60	Continue	clr.l move.l Create lea move move move move move move move	DO, IntBa:  a custom TheScree #0, ns_Le #0, ns_To #320, ns_ #200, ns_De #0, ns_De #1, ns_Bl: #0, ns_Vi	; save base of Intuitions creen	ation library  data structure een
51 ;	Continue	clr.1 move.1 Create lea move move move move move move.b move.b	DO, IntBa:  a custom TheScree #0, ns_Le #0, ns_To #320, ns_ #200, ns_De #0, ns_De #1, ns_Bl: #0, ns_Vi	; save base of Intuination (a); pointer to screen (a); pointer to sc	ation library  data structure een
51 52 53 54 55 56 57 58 59 60 61 62 63	Continue	clr.1 move.1 Create lea move move move move move move.b move move move	DO, Int Base a custom The Scree #0, ns_Le #0, ns_To #200, ns_#200, ns_#200, ns_De #1, ns_Bl #0, ns_Vi #CUSTOMS #0, ns_Fo ScreenTi	; save base of Intuitions are in the screen	ation library  data structure een
51 52 53 54 55 56 57 58 59 60 61 62 63 64	Continue	clr.1 move.1 Create lea move move move move move.b move move move.l lea move.1	DO, IntBar a custom TheScree #0, ns_Le #0, ns_To #320, ns_ #20, ns_De #1, ns_Br #0, ns_Ve #0, ns_Fo ScreenTi A1, ns_De	; save base of Intuitions ; save base of Intuitions ; screen	data structure
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65	Continue	clr.1 move.1 Create lea move move move move move.b move.b move move move move.1 lea move.1	D0,IntBale a custom TheScree #0,ns_Ie #0,ns_Ie #0,ns_Ie #200,ns #200,ns_Bl.ns_	; save base of Intuitive (AO); pointer to screen	data structure
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66	Continue	clr.1 move.1 Create lea move move move move move move.b move move.l lea move.1 lea move.1 move.1	DO, Int Bai  a custom The Scree #0, ns_ Le #0, ns_ Te #0, ns_ De #1, ns_ De #1, ns_ De #1, ns_ Di #CUSTOMS #0, ns_ Fo ScreenTi A1, ns_ De #0, ns_ Ga Int Base,	; save base of Intuitive (AO); pointer to screen (Edge (AO); coordinates of screen (Edge (AO); coordinates of screen (Edge (AO); coordinates of screen (Edge (AO); color for details (Edge (AO); color for rest of (Edge (AO); color for rest of (Edge (AO); color for test of (Edge	data structure
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65	Continue	clr.1 move.1 Create lea move move move move move.b move.b move move move move.1 lea move.1	D0,IntBale a custom TheScree #0,ns_Ie #0,ns_Ie #0,ns_Ie #200,ns #200,ns_Bl.ns_	; save base of Intuitive screen	data structure
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67	Continue	clr.1 move.1 Create lea move move move move move.b move.b move move.1 lea move.1 callsys move.1	DO, Int Base a custom The Scree #0, ns Le #0, ns Le #0, ns Te #200, ns #200, ns #2, ns De #1, ns Ds #1, ns Ds #0, ns Vs #0, ns Fo ScreenTi A1, ns De #0, ns Ga Int Base, OpenScree DO, Scree	; save base of Intuitive control of the control of	data structure een drawing
51 ;	Continue	clr.1 move.1 Create lea move move move move.b move.b move.l move.1 lea move.1 callsys move.1	DO, Int Base a custom The Scree #0, ns_Le #0, ns_To #320, ns_ #20, ns_De #1, ns_Bi #0, ns_Fo ScreenTi A1, ns_De #0, ns_Ge	; save base of Intuitive screen	data structure een  drawing s attached ways come back in D0
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68	Continue	clr.1 move.1 Create lea move move move move.b move.b move.b move.l lea move.1 callsys move.1 callsys move.1	DO, IntBase a custom The Scree #0, ns_Ie #0, ns_To #320, ns_ #2, ns_De #1, ns_Ds #1, ns_Ds #0, ns_To ScreenTi A1, ns_De #0, ns_Ga IntBase, OpenScree DO, Scree  pen a wind The Windo	; save base of Intuitive control of the control of	data structure drawing  s attached ways come back in D0 data structure
51 ; 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 ;	Continue	clr.1 move.1 Create lea move move move move.b move.b move.l move.1 lea move.1 callsys move.1	DO, IntBase a custom TheScree #0, ns_Ie #0, ns_Ie #0, ns_Io #200, ns #200, ns #2, ns_De #1, ns_Bl #0, ns_Vi #CUSTOMS #0, ns_Fo ScreenTi A1, ns_De #0, ns_Ga IntBase, OpenScree pen a wind TheWindo #20, nw_L	; save base of Intuitive control of the control of	data structure een  drawing s attached ways come back in D0
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68	Continue	clr.1 move.1 Create lea move move move move move.b move.b move move.1 lea move.1 callsys move.1 callsys move.1	DO, Int Base a custom The Scree #0, ns_Ie #0, ns_To #320, ns_#2, ns_De #1, ns_B1 #0, ns_To ScreenTi A1, ns_De #0, ns_Ga Int Base, OpenScree DO, Scree  Pen a wind The Windo #20, nw_L #20, nw_T #0, nw_De	; save base of Intuitive control of the control of	data structure drawing  s attached ways come back in D0  data structure coordinates or characters
51 ;	Continue	clr.1 move.1 Create lea move move move move move.b move.b move.1 lea move.1	DO, Int Base a custom The Scree #0, ns Le #0, ns Le #0, ns De #1, ns Ds #200, ns #200, ns Ps #0, ns Vs #0, ns Fo ScreenTi A1, ns De #0, ns Ga Int Base, Open Scree pen a wind The Windo #20, nw L #20, nw T #0, nw De #1, nw Bs	; save base of Intu: screen	data structure drawing  s attached ways come back in D0  data structure coordinates or characters
51 ;	Continue	clr.1 move.1 Create lea move move move move move.b move.l lea move.l lealsys move.l callsys move.l lea move.e move.l bea move.l callsys move.l	DO, IntBar  a custom TheScree #0, ns_Le #0, ns_To #320, ns_ #200, ns_De #1, ns_Bi #0, ns_Fo ScreenTi A1, ns_De #0, ns_Ga IntBase, OpenScree DO, Scree pen a wind TheWindo #20, nw_L #20, nw_T #1, nw_Bl WindowTi	; save base of Intuition  screen	data structure drawing  s attached ways come back in D0  data structure coordinates or characters
51 ;	Continue	clr.1 move.1 Create lea move move move move.b move.b move.l move.1 callsys move.1 callsys move.1 lea move.d move.d move.l	DO, Int Base a custom The Scree #0, ns_Le #0, ns_To #3200, ns_ #2, ns_De #1, ns_Bl #0, ns_To ScreenTi A1, ns_De #0, ns_Ga Int Base, Open Scree DO, Scree DO, Scree DO, Scree DO, Ns_To WindowTi H20, nw_Te H21, nw_Bl	; save base of Intuitive control of the control of	data structure data structure data structure drawing s attached ways come back in D0 data structure coordinates or characters drawing
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 70 71 72 73 74 75	Continue	clr.1 move.1 Create lea move move move move move.b move.l lea move.l lealsys move.l callsys move.l lea move.e move.l bea move.l callsys move.l	DO, Int Bai  a custom The Scree #0, ns _ Ie #0, ns _ Io #320, ns _ #200, ns _ De #0, ns _ De #0, ns _ De #0, ns _ Do #1, ns _ Bi #0, ns _ Fo ScreenTi A1, ns _ De #0, ns _ Go #0, ns _ Go #0, ns _ Go #0, ns _ Go #1, nw _ To #0, nw _ To #0, nw _ To #1, nw _ Bi WindowTi A1, ns _ To #1, nw _ Bi WindowTi A1, ns _ To #WINDOWC	; save base of Intuition  screen	data structure data structure data structure drawing s attached ways come back in D0 data structure coordinates or characters drawing
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 70 71 72 73 74 75 76	Continue	clr.1 move.1 Create lea move move move move.b move.b move.1 lea move.1 move.1 callsys move.1 callsys move.1 lea move move.1 move.1 callsys move.1 move.1 move.1 move.1 callsys move.1	DO, IntBase a custom TheScree #0, ns_Le #0, ns_To #320, ns_ #2, ns_De #1, ns_Bl #0, ns_To ScreenTi Al, ns_De #0, ns_Ga IntBase, OpenScree DO, Scree DO, Scree DO, Scree DO, Ns_To WindowTi #1, nw_De	; save base of Intuitive control of the control of	data structure data structure deen  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  NDOWDRAG+
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 59 70 71 72 73 74 75	Continue	clr.1 move.1 Create lea move move move move.b move.b move.l move.1 callsys move.1 callsys move.1 lea move.d move.d move.l	DO, IntBase a custom TheScree #0, ns_Le #0, ns_To #320, ns_ #2, ns_De #1, ns_Bl #0, ns_To ScreenTi Al, ns_De #0, ns_Ga IntBase, OpenScree DO, Scree DO, Scree DO, Scree DO, Ns_To WindowTi #1, nw_De	; save base of Intu: screen	data structure data structure deen  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  NDOWDRAG+
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 70 71 72 73 74 75 76	Continue	clr.1 move.1 Create lea move move move move.b move.b move.1 lea move.1 move.1 callsys move.1 callsys move.1 lea move move.1 move.1 callsys move.1 move.1 move.1 move.1 callsys move.1	DO, IntBase a custom TheScree #0, ns_Le #0, ns_To #320, ns_ #200, ns_ #2, ns De #0, ns_De #1, ns_Bi #0, ns_To Screen TheWindo #0, nw_L #20, nw_T #0, nw De #1, nw_Bi WindowTi A1, nw_Bi #WINDOWS #CLOSEWII	; save base of Intuitive control of the control of	data structure data structure deen  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  NDOWDRAG+
51 ;	Continue	clr.1 move.1 Create lea move move move move move.b move.b move.1 lea move.1 move.1 callsys move.1 lea move.1 move.b lea move.b lea move.b lea move.l move.l	DO, Int Bai  a custom The Scree #0, ns_Le #0, ns_To #320, ns_ #2, ns_De #1, ns_Bi #0, ns_Fo ScreenTi Al, ns_De #0, ns_Ga Int Base, OpenScree Do, Scree Doen a wind The Windo #10, nw_De #1, nw_Bi WindowTi #20, nw_T #31, nw_Bi WindowTi #WINDOWS #CLOSEWI #CUSTOMS	; save base of Intu: screen	data structure data structure data structure data structure drawing  s attached ways come back in D0 data structure coordinates or characters drawing dinDOWDRAG+ dec.
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 77 78 79 80	Continue	clr.1 move.1 Create lea move move move move move.b move.b move.l move.1 lea move.1 move.1 callsys move.1 lea move.1 move.1 move.1 callsys move.1	DO, Int Base a custom The Scree #0, ns_Le #0, ns_To #320, ns_ #200, ns_ #2, ns_De #1, ns_Bi #0, ns_To ScreenTi A1, ns_De #0, ns_Ga Int Base,, OpenScree DO, Scree pen a wind The Windo #20, nw_T #0, nw_De #1, nw_Bi #1, nw_Bi #CUSTOMS #CLOSEWI #CUSTOMS #CLUSTOMS	; save base of Intuitive control of the control of	data structure data structure deen  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  NDOWDRAG+ cc. )) tted
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 59 70 71 72 73 74 75 76 77 78 79 80 81	Continue	clr.1 move.1 Create lea move move move move move.b move.b move.l move.1 move.1 callsys move.1 lea move move move.l	DO, Int Bai  a custom The Scree #0, ns_Le #0, ns_To #320, ns_ #2, ns_De #1, ns_Bi #0, ns_To ScreenTi Al, ns_De #0, ns_Ga Int Base, OpenScree DO, Scree DO, Scree DO, Scree DO, Ns_To ScreenTi Lat, nw_De H1, nw_De H1, nw_De H1, nw_De H1, nw_De H1, nw_Ti #WINDOWSI #CLOSEWI #CUSTOMS #0, nw_Fi #0, nw_Fi #0, nw_Fi #0, nw_Fi #0, nw_Gh #150, nw_Th	; save base of Intuition  a; ; save base of Intuition  a; ; pointer to screen  Edge (A0) ; coordinates of scr  Edge (A0) ; coordinates of scr  Edge (A0) ; coordinates of scr  Edge (A0) ; color for details  Edge (A0) ; color for rest of  Modes (A0)  Escan ; rupe (A0)  Escan ; rupe (A0)  Escan ; results almost alw  Edge (A0) ; no special gadgets  Edge (A0) ; rupe default font  Edge (A0) ; rupe fact  Edge (A0) ; rupe fact  Edge (A0) ; rupe (A0) ; rupe (A0)  Edge (A0) ; rupe (A0) ; rupe (A0)  Edge (A0) ; rupe (A0) ;	data structure data structure deen  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  NDOWDRAG+ cc. )) tted
51 ;	Continue	clr.1 move.1 Create lea move move move move move move.b move.b move.1 lea move.1 move.n	DO, Int Bai  a custom The Scree #0, ns _ Ie #0, ns _ To #320, ns _ #200, ns _ De #1, ns _ Bi #0, ns _ Fo #1, nw _ Fo #0, nw _ Fo #1, nw _ Fi #0, nw _ Fi #150, nw _ Fi #280, nw _ Fi	; save base of Intu: screen	data structure data structure deen  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  ENDOWDRAG+ cc. c) tted dgets attached menu items
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 59 70 71 72 73 74 75 76 77 78 79 80 81	Continue	clr.1 move.1 Create lea move move move move move.b move.b move.l move.1 lea move.1 move move move move move move move move	DO, Int Base a custom The Scree #0, ns_Le #0, ns_To #320, ns_ #20, ns_De #1, ns_Bi #0, ns_To ScreenTi A1, ns_De #0, ns_Ga Int Base, OpenScree DO, Scree DO, Scree Pen a wind The Windo #20, nw_L #20, nw_T #1, nw_Bi #CUSTOMS #CLOSEWI #CUSTOMS #CLOSEWI #CUSTOMS #CLOSEWI #150, nw_Ch #150, nw_H #280, nw_H #280, nw_H #280, nw_H #280, nw_H #100, nw_Ch #150, nw_H #150, nw_H #100, nw_H	; save base of Intuitive control of the control of	data structure data structure deen  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  ENDOWDRAG+ cc. c) tted dgets attached menu items
51 ;	Continue	clr.1 move.1 Create lea move move move move move move.b move.b move.1 lea move.1 move.n	DO, Int Base a custom The Scree #0, ns_Le #0, ns_To #320, ns_ #2, ns_De #1, ns_Bi #0, ns_To ScreenTi Al, ns_De #0, ns_Ga Int Base, OpenScree DO, Scree DO, Scree DO, Scree DO, Ns_To WindowTi #10, nw_To WINDOWS #CLOSEWI #CUSTOMS #CLOSEWI #CUSTOMS #CUSTOMS #CLOSEWI #CUSTOMS #0, nw_To #150, nw_To #150, nw_To #150, nw_To #150, nw_To #150, nw_To #100, nw_To	; save base of Intu: screen	data structure data structure deen  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  ENDOWDRAG+ cc. c) tted dgets attached menu items
51 ;	Continue	clr.1 move.1 Create lea move move move move move.b move.b move.l move.1 lea move.1 move.1 move.1 move.1 move.1 move.1 move.1 move.1 move.1 move move move move move move move move	DO, Int Base a custom The Scree #0, ns_Le #0, ns_To #320, ns_ #20, ns_De #1, ns_Bi #0, ns_To \$1, ns_Bi #CUSTOMS #0, ns_Fo ScreenTi A1, ns_De #0, ns_Ga Int Base, OpenScree DO, Scree DO, Scree Poen a wind The Windo #10, nw_De #1, nw_Bi #0, nw_To #0, nw_To #0, nw_To #0, nw_Fi #0, nw_Fi #0, nw_Fi #10, nw #280, nw #150, nw #150, nw #150, nw #1640, nw #1640, nw #1640, nw #1640, nw #1200, nw	; save base of Intuitive control of the control of	data structure data structure deen  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  ENDOWDRAG+ cc. c) tted dgets attached menu items
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 77 78 79 80 81 82 83 84 85 86 87	Continue	clr.1 move.1 Create lea move move move move move.b move.b move.l move.1 move.b move move.b move move.b move move.b move move move.b move.b move.b move.b move.b move.l move.l move.l	DO, Int Base a custom The Scree #0, ns_Le #0, ns_To #320, ns_ #20, ns_ #2, ns_De #1, ns_Bl #0, ns_To ScreenTi Al, ns_De #0, ns_Ga Int Base,, OpenScree DO, Scree DO, Scree DO, Scree The Windo #20, nw_T #0, nw_De #1, nw Bl #10, nw_Ti #WINDOWSI #CLOSEWII #CUSTOMS #CUSTOMS #CUSTOMS #0, nw_Fi #0, nw_Fi #0, nw_Fi #150, nw #280, nw #150, nw #150, nw #100, nw	; save base of Intuition  a ; pointer to screen  Edge (A0) ; coordinates of scr  Edge (A0) ; color for details  ExPen (A0) ; color for rest of  Modes (A0)  ESEN, ns Type (A0)  ES (A0) ; use default font  EL, A1  Edge (A0) ; no special gadgets  Etr ; results almost alv  Expen (A0) ; color for rest of  EL, A1  EL (A0)  EL (A0)  ESED, MART REFRESH+ACTIVATE+WI  ENGHNINDOWDEPTH, nw Flags (A0)  EXPEN (A0) ; color for rest of  ELEN, nw Type (A0)  ESEN, nw Type (A0)  ESEN, nw Type (A0)  ESEMARY (A0) ; no special gad  ESEN, nw Type (A0)  ESEMARY (A0) ; no special gad  ESEN, nw Type (A0)  ESEMARY (A0) ; no special gad  ESEN, nw Type (A0)  ESEMARY (A0) ; initial  ENGHNING (A0)  ENGHNING	data structure data structure deen  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  ENDOWDRAG+ cc. c) tted dgets attached menu items
51 ;	Continue	clr.1 move.1 Create lea move move move move move.b move.b move.l move.1 lea move.1 move.1 move.1 move.1 move.1 move.1 move.1 move.1 move.1 move move move move move move move move	DO, Int Base a custom The Scree #0, ns_Le #0, ns_To #320, ns_ #20, ns_De #1, ns_Bi #0, ns_To \$1, ns_Bi #CUSTOMS #0, ns_Fo ScreenTi A1, ns_De #0, ns_Ga Int Base, OpenScree DO, Scree DO, Scree Poen a wind The Windo #10, nw_De #1, nw_Bi #0, nw_To #0, nw_To #0, nw_To #0, nw_Fi #0, nw_Fi #0, nw_Fi #10, nw #280, nw #150, nw #150, nw #150, nw #1640, nw #1640, nw #1640, nw #1640, nw #1200, nw	; save base of Intuitive control of the control of	data structure een  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  NDOWDRAG+ cc. )) tted ligets attached menu items a be sized
51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 86 86 86 87 87 87 87 87 87 87 87 87 87	Continue	clr.1 move.1 Create lea move move move move move.b move.b move.l move.1 move.b move move.b move move.b move move.b move move move.b move.b move.b move.b move.b move.l move.l move.l	DO, Int Base a custom The Scree #0, ns_Le #0, ns_To #320, ns_ #20, ns_ #2, ns_De #1, ns_Bl #0, ns_To ScreenTi Al, ns_De #0, ns_Ga Int Base,, OpenScree DO, Scree DO, Scree DO, Scree The Windo #20, nw_T #0, nw_De #1, nw Bl #10, nw_Ti #WINDOWSI #CLOSEWII #CUSTOMS #CUSTOMS #CUSTOMS #0, nw_Fi #0, nw_Fi #0, nw_Fi #150, nw #280, nw #150, nw #150, nw #100, nw	; save base of Intuitive control of the control of	data structure data structure deen  drawing  s attached ways come back in D0  data structure coordinates or characters drawing  ENDOWDRAG+ cc. c) tted dgets attached menu items



#### WRITE BETTER C PROGRAMS.

#### C TOOLS PLUS

Programmer's Connection is pleased to present *C TOOLS PLUS*, one of the most complete *C* language function libraries available today. It combines the general purpose capabilities of *C TOOLS* and *C TOOLS* 2 (two packages that receive rave reviews for quality, organization, usability and documentation) with extensive new windowing capabilities.

#### Windows

For the first time, you can get full functioned window support in a general purpose library. The routines are easy to use for pop-up menus or windows and can give you some very advanced capabilities. Windows can be stacked, removed and can accept user input. There is no limit, other than available dynamic memory, to the number of windows you can construct and use.

#### **Video Support**

C TOOLS PLUS has fast direct video access for efficiency that will not constrain good program design. You can write directly to the video adapter for efficiency or use only BIOS calls if you're working in a TopView-like environment. It fully supports EGA text modes including 43-line mode and multiple display pages.

#### **Memory-resident DOS Access**

Blaise Computing's C TOOLS 2 package is well-known for effectively allowing you to write interrupt service routines in C. C TOOLS PLUS takes these capabilities even further. It demonstrates how to access DOS functions from within an interrupt service routine, so now you can write memory-resident routines that access the disk.

#### **General Purpose Functions**

C TOOLS PLUS consists of over 200 carefully written functions that are easy-to-use and supplied in source code and prebuilt library form. These tools isolate hardware dependence, are small and are written predominently in C.

Some of the areas covered are: ● extensive string handling ● screen handling ● graphics interface ● general utility and keyboard functions ● DOS memory management ● external program and DOS internal command execution ● DOS file handling and directory maintenance ● and much more.

All Blaise Computing products come with complete, fully documented source code as well as a complete comprehensive, indexed reference manual.

#### Requirements

C TOOLS PLUS requires an IBM personal computer (PC/XT/AT) or compatible, although many of the functions require only MS—DOS Version 2.00 or later. C TOOLS PLUS supports the Microsoft and Lattice C compliers (versions 3.00 or later)

#### **Other Products**

Other quality products for C from Blaise Computing include:

ASYNCH MANAGER — provides the crucial core of hardware interrupt support needed to build applications that need asynchronous communication facilities. It also includes routines for XMODEM file-transfer protocol and support for Hayescompatible modems.

VIEW MANAGER — a development system for building data entry screens and menus. Begin by designing on-screen what the operator will see, then call upon library functions from your program to display the screens and retrieve data.

#### BLAISE COMPUTING INC.

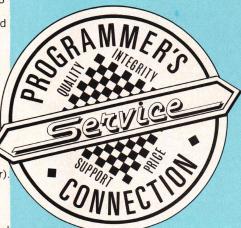
Circle no. 98 on reader service card.

#### YOUR QUALITY CONNECTION...

Blaise Computing products are available from Programmer's Connection at these low discount prices with FREE UPS shipping for all US customers (express services are available at the shipping carrier's standard rate with no extra rush fees).

C TOOLS PLUS	List 175	<b>Ours</b> 135
ASYNCH MANAGER Specify C or MS Pascal C TOOLS	175 125	135 99
C TOOLS 2 PASCAL TOOLS for MS Pascal	100 125	79 99
PASCAL TOOLS 2 for MS Pascal PASCAL TOOLS & PASCAL TOOLS, 2	100 175	79 135
TURBO ASYNCH PLUS for Turbo Pascal TURBO POWER TOOLS PLUS Turbo Pascal	100 100	83 83
VIEW MANAGER Specify C or MS Pascal	275	199

Please refer to our main advertisement in this journal for more information about our services as well as the largest advertised selection of programmer's development tools specifically for IBM personal computers and compatibles.



800-336-1166 800-225-1166 OHIO AND OVERSEAS 216-877-3781 CUSTOMER SERVICE

programmer's connection

136 SUNNYSIDE ST. HARTVILLE, OHIO 44632

#### **MULTITASKING**

Introducing

#### **MultiDos Plus**

The new multitasking software for the IBM-PC.

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in any language.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/ resume programs.
- Programmatic interface via INT 15H for the following.
  - Intertask message communication. Send/receive/check message present on 64 message queues.
  - \* Task control by means of semaphores. Get/release/check semaphores.
  - Change priority-128 priority levels.
  - Suspend task for specified interval.
  - Spawn and terminate external and internal tasks.
  - \* Disable/enable multitasking.
  - \* and more!
- Runs most commodity software including: Lotus 1-2-3, Dbase, Wordstar, and others.
- Independent foreground / background displays.
- Access to DOS while applications are running.

#### Hardware/Software Requirements

IBM PC/XT/AT or true clone. Monochrome/CGA display adaptors or equivalent cards only. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

ONLY

\$29.95 +\$2.95 S/H

Outside USA add \$7.95 shipping and handling. Visa and Mastercard orders call toll-free: 1-800-367-6707. In Mass call 617-651-0091, or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760

MA orders add 5% sales tax. Write for source code and quantity price.

Circle no. 309 on reader service card

#### MAC AND AMIGA

#### Listing Three (Listing continued, text begins on page 40.)

89 90 91	callsys lea move.l	OpenWindow WindowPtr,A0 D0,(A0)	
; Se 92 93 94 95 96 97 98	passtext passtext passtext passtext passtext passtext	menus ProjText1, Proj1 ProjText2, Proj2 ProjText3, Proj3 ProjText4, Proj4 ProjText5, Proj5 ProjText6, Proj6 ProjText7, Proj7	;First must initialize all ;Intuition text structures.
99 100 101 102 103 104 105 106 107 108 109 110	lea passitem lea passitem lea passitem lea passitem lea passitem	ProjItem2, A3 ProjItem1, A3, ProjTe ProjItem2, A3, ProjTe ProjItem4, A3 ProjItem4, A3 ProjItem5, A3 ProjItem4, A3, ProjTe ProjItem6, A3 ProjItem6, A3, ProjTe ProjItem6, A3, ProjItem6, A3, ProjItem6, A3, ProjItem6, A3, ProjItem6, A3, ProjItem7, A3 ProjItem6, A3, ProjTe ProjItem6, A3, ProjTe	<pre>;in menu item structures. xt2,#'0',#9 xt3,#'s',#18 xt4,#'A',#27 xt5,#'P',#36 xt6,#'R',#45</pre>
112 113 114 115 116 117 118 119 120 121 121 122 123	lea lea move.l move move move move lea move.l lea move.l	#0, mu_LeftEdge (A0) #0, mu_TopEdge (A0) #0, mu_Height (A0) #100, mu_Width (A0) #MENUENABLED, mu_Fla ProjName, Al Al, mu_MenuName (A0) ProjItem, Al	;Finally, must initialize the ;menu structure itself. ;pointer to next menu in list ;place for title in menu strip ;ignored ;ignored ags(AO) ;menu is enabled );head of menu item list
124 125 126 127 128	passtext passtext passtext	EditText1, Edit1 EditText2, Edit2 EditText3, Edit3 EditText4, Edit4 EditText5, Edit5	; Now, repeat process for ; the second menu.
129 130 131 132 133 134 135 136 137	lea passitem lea passitem lea passitem	EditItem2, A3 EditItem1, A3, EditTe EditItem3, A3 EditItem2, A3, EditTe EditItem4, A3 EditItem3, A3, EditTe EditItem5, A3 EditItem4, A3, EditTe EditItem5, A5, EditTe EditItem5, B6, EditTe	ext2,#'X',#9 ext3,#'C',#18 ext4,#'V',#27
138 139 140 141 142 143 144 145 146 147	lea move.l move move move lea move.l lea move.l	#0, mu NextMenu (A0) #101, mu LeftEdge (A) #0, mu TopEdge (A0) #75, mu Width (A0) #0, mu Height (A0) #MENUENABLED, mu FlateditName, A1 A1, mu MenuName (A0) EditItem1, A1 A1, mu FirstItem (A0)	;end of the list 0) ags(A0)
149 150 151 152	move.l move.l lea callsys	IntBase,A6 WindowPtr,A0 ProjMenu,A1 SetMenuStrip	;window in question ;first menu in strip ;attach menu strip to window
153 154 155	lea lea move.l jsr	message ports for co WritePort,A3 #0,A5 CreatePort	nsole; storage for pointer to write port; unnamed ports - first in list; initialize the port
156 157 158	move.l lea jsr	WritePort,A3 WriteMsg,A5 CreateStdIO	;write port pointer ;storage for pointer to IO block ;initialize IO block
159 160 161	lea lea jsr	ReadPort, A3 ReadName, A5 CreatePort	Repeat for read port.; has name - not first in list
162 163 164	move.l lea jsr en and att	ReadPort, A3 ReadMsg, A5 CreateStdIO aCh the console devi	
165 166	move.l move.l	ach the console devi WriteMsg,A3 ReadMsg,A5	;output IO request block ;input IO request block

of the second second		# 100 No. 100 No. 100	Charles State of the Control of the	
267		morro 1	WindowPtr A/	window for this console
167 168		isr	OpenConsole	;window for this console
169		amp	#0,D0	
170		beq		1443
171		rts		;unsuccessful opening ends program
172 173		move 1	WindowPtr,A0 wd UserPort (A0) .A0	;message port for Intuition
174		move.b	MP SIGBIT (A0), D0	;Intuition signal bit
175		lea	IntSigBit,A0	
176		move.b	DO, (AO)	;Intuition signal bit ;save it
177		move.1	MB STCRITT (AC) DO	.concolo gignal hit
178 179		lea	Considert AO	; console signal bit
180		move.b	DO, (AO)	; save it
:	Qu	eue up an	initial read reques	st
181 182		move.1	letter MA	console IO request block; place to put character read
183		dsr	letter,A4 QueueRead	; queue up a message
<i>i</i>				ent
184 185	Event	move.b	IntSigBit,D1	
186			DO DO	
188		bset.1	D1,D0	; sys will look for Intuition event
189		clr.l	D1	
190		move.b	ConSigBit,D1	;system looks for console evt, too
191 192		move.1	D1,D0 AbsExecBase,A6	, system tooks for consore eve, too
193		callsys	Wait	;wait for event to occur
194		clr.l	D1 IntsigBit D2	; Note - now a bit in DO is set
195 196		bset 1	IntSigBit,D2	to correspond to signal causing ;event.
190		amp.1	IntSigBit,D2 D2,D1 D1,D0	;Incuition event?
198		beq	IntuitionEvent	
199		clr.l	D1 ConcieDit D2	
200		hset 1	ConSigBit,D2	
202		cmp.1	D2,D1 D1,D0	; console event?
203		beq	ConsoleEvent	
204			-	-6-416- b leben14 ben-
204		bra	Event	;fail-safe trap-should never get here
;	Han	ndle Intui	tion events	
	Intuition			
205		move.l	WindowPtr,A0	. Intuition is masses as w
206		move.1	AbsExecBase A6	;Intuition's message port
208				;retrieve the input message
209		beq		; no message present
210		morro 1	DO 31	*CotMag roturns address of mossage in DO
210		move.1	im Class(A1),D0	;GetMsg returns address of message in DO ;type of event
212		amp	#CLOSEWINDOW, DO	; was window close box clicked?
213		beq	CloseAndQuit	
01.4			"MONTO TOK DO	.many abadaa mada2
214 215		beq	#MENUPICK, DO MenuEvent	;menu choice made?
210		Joq		
	DoneWithE			
216		move.l	AbsExecBase, A6	Annance magazine as it was be married
217		callsys	ReplyMsg Event	remove message so it can be reused
210		MIG	Lveiic	
	MenuEvent			
219		move	im_Code(A1),D0	;menu & menu item number
220		beq	DoneWithEvent	;user backed out before chosing
221		move	D0,D1	;save the code
222		and	#%0000000000011111	,DO ;get menu number
223		amp	#0,D0	;project menu?
224		bne	DoneWithEvent	;Project menu is the only one
100				trapped by this program!!!!
225		lsr	#5,D1	
226		and		,DO ;get menu item number
227		amp	#6,D1	;Quit selected?
228		bne	DoneWithEvent	;Quit is the only option ;implemented by this program!!!!!
				,
	CloseAndQ			
229		move.l	ReadMsg, A1	Trameric last event from guara
230		ABORTIO		; remove last event from queue
231		move.1	ReadMsg, Al	
1 401		move.1	AbsExecBase, A6	
232			CloseDevice	; close the console
232 233		callsys	CloseDevice	
233				
233		move.1	WindowPtr,A0	
233				;close the window
233 234 235		move.l	WindowPtr,A0 IntBase,A6	

#### Fortran Support

for

IBM PC/XT/AT & Compatibles

Versions Available For:

Microsoft, Supersoft, RyanMcFarland, IBM Professional, Lahey, & IBM Fortran.

#### Forlib-Plus \$69.95

Supports graphics, interrupt driven communication, program chaining, and file handling/ disk support. A Fortran coded subroutine is included which will plot data on the screen either in linear/linear, log/ linear, linear/log, or log/log on the appropriate grid.

#### Strings & Things \$69.95

Supports string maipulations, command line usage, DOS call capabilities, SHELL generation and data transmission, BATCH file control, music generation, PEEKS and POKES, PORT access, and general register manipulations.

#### For-Winds \$89.95

Gives the Fortran programmer the capability of generating up to 255 windows on the screen. Each window can be individually scrolled, moved, sized, generated, and removed. Both color and monochrome type displays are supported. Full source code is supplied for customization.

#### **ACS Time Series** \$495.00

This is a COMPLETE time series analysis package which contains VERY HIGH SPEED FFTs, Filter generations, convolutions, transfer function calculations, auto and cross spectra calculations, Cepstrum, curve fitting algorithims, coherence calculations, and many other associated routines. The price includes FULL source code.

#### Fortran Scientific **Subroutine Package** \$295.00

There are approximately 100 Fortran subroutines included which fall under the following 12 categories:

1) Matrix storage and Operations 2) Correlation and Regression, 3) Design Analysis (ANOVA), 4) Descriminant Analysis, 5) Factor Analysis, 6) Eigen Analysis, 7) Time Series, 8) Nonparametric Statistics, 9) Distribution Functions, 10) Linear Analysis, 11) Polynomial Solutions, 12) Data Screening. Full source code is included.



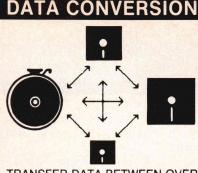
ALPHA COMPUTER SERVICE 5300 ORANGE AVENUE SUITE 108 CYPRESS; CALIFORNIA 90630 (714) 828-0286

California Residents

Include 6% Sales Tax

There are NO license fees

Circle no. 321 on reader service card.



TRANSFER DATA BETWEEN OVER 600 DIFFERENT COMPUTER SYSTEMS

WORD PROCESSORS TOO QUICK TURN-AROUND

PRICES FROM \$9 PER DISK CALL OR WRITE FOR YOUR

FREE CATALOG PORT-A-SOFT

555 S. STATE ST., SUITE #12 P.O. BOX 1685, OREM, UT 84057 (801) 226-6704

Circle no. 229 on reader service card.

#### SOURCE CODE LIBRARIAN & REVISION CONTROL SYSTEM

TLIB™ keeps ALL versions of your program in ONE compact library file, eveň with hundreds of revisions!

- Super Fast! Updates (deltas) average 5-7 times faster than PC/IX (Unix) SCCS. TLIB updates libraries faster than many editors load and save files!
- · LAN-compatible! Shared libraries with PC Network!
- · Synchronized control of multiple related source files
- · Use with floppies or hard disk. TLIB doesn't need big bee with hoppines of hard disk. This doesn't need ong temporary files, so you can maintain a 300K library on one 360K diskette, with room to spare, even with TLIB itself on the same disk. And libraries are more compact than with most other revision management systems
- Perfect for backup. Date and comments kept with each version. High data integrity because library data, once written, is never modified. Libraries are only appended, to minimize the chance of data loss due to a power glitch or hardware crash. And TLIB isn't copy-protected, either.
- Free copy of Landon Dyer's excellent public domain MAKE utility. With macros, full source code. For DOS & VAX/VMS.

PC/MS-DOS 2.x & 3.x Just \$99.95 + \$3 s/h Visa/MC

#### **BURTON SYSTEMS SOFTWARE**

P. O. Box 4156, Cary, NC 27511-4156 (919) 469-3068

Circle no. 212 on reader service card

#### MAC AND AMIGA

#### Listing Three (Listing continued, text begins on page 40.)

237	move.1	ScreenPtr, A0	
238	move.1	IntBase, A6	
239	callsys	CloseScreen	; close the custom screen
240	rts		;return to DOS
; Ha:	ndle console	e events	
Console	Event		
241	move.1	ReadMsg, A0	
242	move.1	AbsExecBase, A6	
243	callsys		;retrieve the message
244	move.1	WriteMsg, Al	;output IO request block
245	lea	letter, A4	;place where character is stored
246	jsr	ConPutChar	;display the character
247	jsr move.b cmp	letter,D0	
248	CMD	#\$D,D0	;was character a <cr>?</cr>
249	bne	MoreLetters	, was character a terr.
250	morro h	#\$A, (A4)	; put line feed in letter
251			, put line leed in letter
252		WriteMsg, Al	add a line feed to the <cr></cr>
232	jsr	ConPutChar	; add a line leed to the <cr></cr>
MoreLet			
253		ReadMsg, Al	
254		DevAdd, IO DEVICE (A	
255	jsr	QueueRead ; go get a	another letter
256	bra	Event	
; Subro	utine to loa	ad Intuition text st	ructures
257 SetText	move.b	#FrontPen, it Front	Pen(AO) ; colors for drawing
258	move.b	#BackPen, it BackPen	n (AO)
259	move.b	#0, it DrawMode (A0)	
260	move	#2.it LeftEdge (AO)	:posn rel to container
261	move	#1.it TopEdge (A0)	;posn rel to container
262	move.1	#0.it ITextFont (A0)	):use default font
263	move.1	A1. it TText (A0)	;use default font ;pointer to the text structure
264	move.1	#0. it NextText (A0)	;no link to other txt structs
265	rts	"o, It_Italierane (no,	71.0 11.11 00 00.101 01.0 101.101
203	103		
; Sub	routine to 1	oad menu item struc	tures
	n move.1		;pointer to next item in list
267 Secretary	move.1		;posn rel to container
			posit fer to container
268	move	D1, mi TopEdge (A0)	
269	move	#100, mi_Width (A0)	
270	move	#9,mi_Height (A0)	
271	move	#ITEMTEXT+COMMSEQ+	ITEMENABLED+HIGHCOMP, mi_Flags(A0)
272	move.1	#0, mi_MutualExclude	e (AO) ; no mutually exclusive items
273	move.1	A2, mi_ItemFill(A0)	e (AO) ; no mutually exclusive items ; pointer to text structure
274	move.b	DO, mi_Command (AO)	; keyboard equivalent; no subitems
275	move.l	#0,mi_SubItem(A0)	;no subitems
276	move	#0, mi NextSelect (A	0) ;no associated items
277	rts		
; Cr	eate a messa	age port	
			version of the C source
			It needs error trapping
; after the system			
		message port struct	ure in A3.
		essage port in A5.	
		7	( .: - J maga 70)

(continued on page 79)

#### Here's why you should choose Periscope as your debugger...

You'll get your programs running fast. "It works great! A problem we had for three weeks was solved in three hours," writes Wade Clark of MPPi, Ltd.

You'll make your programs solid. David Nanian says, "I can't live without it!! BRIEF, a text editor my company wrote, would not be as stable as it is today without Periscope."

You'll protect your investment. We won't forget you after the sale. You'll get regular software updates, including a FREE first update and notice of later updates. You'll get technical help from Periscope's author. And you'll be able to upgrade to more powerful models of Periscope if you need to. One Periscope user writes, "...

your support has won over even the heart of this hardened programmer!"

You deserve the best. Thousands of programmers rely on the only debugger that PC Tech Journal has ever selected as Product of the Month (1/86). You owe it to yourself to find out why,

You can try it at no risk. You get an unconditional 30-Day, Money-Back Guarantee, so you can't lose

Start saving time and money now - order tollfree, 800/722-7006. Use MasterCard, Visa, COD, or a qualified company purchase order. As one user puts it, Periscope is "one of the rare products, worth every penny!"

Periscope I, software, manual,	
protected memory board and	
breakout switch	\$295
Periscope II, software, manual, and	
breakout switch	\$145
Periscope II-X,	
software and manual	\$115

Add shipping - \$3 US; \$8 Canada; \$24 elsewhere. Ask about air shipment if you can't wait to get your programs up and running!

#### **PERISCOPE**

#### The Periscope Company, Inc.

(formerly Data Base Decisions) 14 Bonnie Lane, Atlanta, GA 30328 404/256-3860 4 REASONS TO CHOOSE PROGRAMMER'S CONNECTION:

# QUALITY SUPPORT PRICE & INTEGRITY

As we enter 1987, we'd like to extend a warm thank you to our customers and wish everyone a Happy New Year!

If you've bought from us before, we look forward to serving you again. And if you're not yet familiar with our one-stop service, we invite you to give us a call.

It's our commitment to quality, support, low prices and integrity that makes us your best source for the programming tools you need. So make the connection today and discover the value and convenience of our one-stop service for yourself. You'll be glad you did!

We carry the finest selection of the best programmer's development tools specifically for IBM Personal Computers and compatibles. They are the latest versions and most come with 30-day documentation evaluation periods or 30-day return guarantees.

We firmly believe that high quality must be present throughout every aspect of our service. So to make sure that we maintain such high standards, we include a service questionnaire with every purchase. We're very interested in what our customers have to say.

Our courteous, knowledgeable, noncommissioned salespeople are always ready to assist you. We also have experienced technical consultants on staff who can answer questions about products and provide sound, unbiased advice. We'll support you before and after you make your purchase. Your satisfaction is very important to us.

Our buying power enables us to offer you the lowest prices without sacrificing service. UPS shipping is FREE to all U.S. customers. There are no extra charges for credit cards, CODs, purchase orders or special handling (except for export preparation).

Quite simply, the discount prices listed with the products on the next two pages are all you pay. There are no hidden add-on charges.

When we started Programmer's Connection in 1984, we dedicated ourselves to providing high quality personal service to every customer. Since then, we've quickly grown to be the leading independent dealer in this industry.

**W**e're very proud of the trust we've earned from our customers and we pledge always to be worthy of it.

Call Toll Free United States 800-336-1166 Canada 800-225-1166 Ohio & Overseas 216-877-3781



Turn the page for our product listing and ordering information.

**Circle Reader Inquiry Number 129** 

apl language			Turbo EDITOR TOOLBOX	70	48	FORMS-2 300 259
APL*PLUS/PC by STSC	595	429	Turbo GAMEWORKS TOOLBOX	70		Level II Animator 900 349
APL*PLUS/PC Spreadsheet Mgr by STSC .	195	139	Turbo GRAPHIX TOOLBOX	70	48	Level II SOURCEWRITER 2000 CALL
APL*PLUS/PC Tools Vol 1 by STSC	295	199	Turbo LIGHTNING	100	65	Micro Focus Level II COBOL for Novell 2000 1699
APL*PLUS/PC Tools Vol 2 by STSC	85	59	Turbo PASCAL with 8087 and BCD	100	65	Micro Focus Micro/SPF 175 149
APL*PLUS/UNX For AT XENIX by STSC	995	695	Turbo Prolog Compiler	100	65	Micro Focus Professional COBOL 3000 2295
Btrieve ISAM File Mgr by SoftCraft	245	194	Turbo TUTOR for Turbo PASCAL	40	28	Multi-user Runtime for PC Network 500 429
Financial / Statistical Library by STSC	275	195	Word Wizard	70	48	Microsoft COBOL Compiler 700 439
Pocket APL by STSC	95	69	Word Wizard and Turbo Lightning	150	99	for XENIX
STATGRAPHICS by STSC	795	579	c++			Realia COBOL
artificial intelligenc	_		C++ from Guidelines New Version	195	179	RM/COBOL by Ryan-McFarland 950 639 RM/COBOL 8X ANSI 85
#####################################			C 1 1 nom addennes	100	173	by Ryan-McFarland
1st-CLASS by Programs in Motion	495	399	c compilers			
APT from Solution Systems		CALL	C86PLUS by Computer Innovations New	497	CALL	debuggers & profilers
	CALL		Datalight C Compiler Small Model	60		386 DEBUG Cross Debugger by Phar Lap 195 159
AutoIntelligence by IntelligenceWare ESP ADVISOR by Expert Systems Intl	895	CALL	Datalight Developer Kit w Large Model	99	79	Advanced Trace-86 by Morgan Computing 175 125
PROLOG-2 Interface	395	839 369	DeSmet C w Debugger	159	145	CI Probe by Computer Innovations 225 189
ExpertEDGE Advanced by Human Edge		CALL	DeSmet C w Debugger & Large Case	209	193	Codesifter Profiler by David Smith 119 98
ExpertEDGE Professional by Human Edge		CALL	Eco-C Development System by Ecosoft	125	89	Codesmith-86 by Visual Age 145 108
Experteach II by Intelligence Ware	475	359	Lattice C Compiler from Lattice	500		DSD86 by Soft Advances 70 65
EXSYS Development Software by EXSYS	395	319	Mark Williams Let's C	75	58	DSD87 by Soft Advances 100 89
GCLISP Golden Common LISP by Gold Hill		CALL	with csd Source Debugger	150		Periscope I by The Periscope Company 295 245
	1190	CALL	Mark Williams MWC-86	495		Periscope II w NMI Breakout Switch 145 109 Periscope II-X Software only 115 84
Insight 1 by Level Five Research	95	75	Microsoft C with CodeView	450	275	The PROFILER with Source Code by DWB 125 94
Insight 2+ by Level Five Research	485	379	Cross Compiler	595	CALL	The WATCHER Profiler by Stony Brook 60 55
Intelligence / Compiler Intelligence Ware	990	749	Wizard C Combo by Wizard Systems	750		
Logic-Line Series 1 by Thunderstone	90	85	Wizard C Compiler	450		forth language
Logic-Line Series 2 by Thunderstone	125	115	ROM Development Pkg	350		CFORTH Native Code Compiler by LMI 300 239
Logic-Line Series 3 by Thunderstone  LPA microPROLOG by Prog Logic Systems	150	139				Forth/83 Metacompiler Specify Target
with APES	149	89 129	cinterpreters			PC/Forth by Laboratory Microsystems 150 119
LPA Professional microPROLOG	395	339	C-terp by Gimpel, Specify compiler		235	PC/Forth + by Laboratory Microsystems 250 209
with APES	650	569	C Trainer with Book by Catalytix		CALL	Advanced Color Graphics Support 100 79
Microsoft LISP Common LISP	250	169	Instant C by Rational Systems		CALL	Enhanced Graphics Support 200 159
PC Scheme by Texas Instruments	95	85	Introducing C by Computer Innovations	125		Intel 8087 Support 100 79
Personal Consultant Easy by Tl	495	439	Run/C from Lifeboat	150	89	Interactive Symbolic Debugger 100 79
	2950		Run/C Professional from Lifeboat	250	169	Native Code Optimizer 200 159
Personal Consultant Runtime	CALL	CALL	c utilities			PCTERM Modem Pgm for Smartmodem 100 79
PROLOG-2 Interpreter by ESI	450	419	See also Blaise, GSS, Lattice, Microsoft, P	hooni		Software Floating Point
PROLOG-2 Interpreter and Compiler	895	839	Polytron, SoftCraft and XENIX sections.	noem	^,	UR/Forth by Laboratory Microsystems New 350 279
QNIAL by NIAL Systems	375	349		205	200	Object Module Libraries New 500 395
TransLISP from Solution Systems		CALL	APT by Shaw American Technology Basic C Library by C Source	395 175		Source Code License New 1500 995
Turbo PROLOG Compiler by Borland Intl	100	75	C Essentials by Essential Software		CALL	fortran language
assembly language			C-ISAM by Informix	225		ioi ti ali laliguage
386 ASM/LINK Cross Asm by Phar Lap	105	CALL	C to dBase by Computer Innovations	150		50 MORE: FORTRAN by Peerless Engr 125 99
8088 Assembler w Z-80 Trans by 2500 AD .	100	89	c-tree & r-tree Combo Package New	650		ACS Time Series Alpha Computer Service 495 419
ASMLIB Function Library by BC Assoc	149	129	c-tree ISAM File Manager by FairCom	395	329	Btrieve ISAM File Mgr by SoftCraft 245 194
asmTREE B-Tree Dev System by BC Assoc	395	339	r-tree Report Generator	295	249	Essential Graphics by Essential Software 250 195
	CALL	CALL	C Utility Library by Essential Software	185	135	For-Winds Alpha Computer Service 90 78
Microsoft Macro Assembler	150	98	C Windows by Syscom	100	89	Forlib-Plus Alpha Computer Service 70 54
Norton Utilities by Peter Norton	100	59	C Wings by Syscom	50	45	FORTLIB by The Librarian
Turbo EDITASM by Speedware	99	84	CI ROMPac by Computer Innovations		CALL	FORTRAN Addendum by Impulse Engr 165 149
UniWare Cross Assemblers Various New		CALL	dbQUERY All Varieties by Raima		CALL	GRAFLIB by The Librarian 175 CALL
Visible Computer: 8088 Software Masters	80	65	dbVISTA Single-User DBMS by Raima	195		HALO by Media Cybernetics 300 209
basic language			with Source Code	495 495		I/O PRO w No Limit Library by MEF 390 349
[12] ' 사용한 사용한 위원 경영 불편의 출경 (12) (12) (12) (13) (14) (15) (15) (16) (16) (16) (16) (17) (17) (17) (17) (17) (17)	200	120	dbVISTA Multi-User DBMS by Raima with Source Code	990		Microcompatibles Combo Package 240 219
BetterBASIC by Summit Software	200	129 75	dBx dBase C Translator by Desktop Al	350		Grafmatic
Btrieve Interface	99	75	with Library Source Code	550		Plotmatic 135 119
C Interface	99	75	Entelekon Combo Package	200		Microsoft FORTRAN Compiler 350 204
Run-time Module	250	169	C Function Library	130		No Limit by MEF Environmental 129 115
EXIM Services Toolkit by EXIM New			C Windows	130		PANEL Screen Designer by Roundhill 295 224
Finally by Komputerwerks New	99	85	Superfonts for C	50	43	PLOTHI by The Librarian
Inside Track from Micro Help	65	55	Essential Comm Library			PLOTHP by The Librarian
MACH 2 by Micro Help	75	65	with Debugger New	250		Sci Subroutine Library by Peerless 175 138
Microsoft QuickBASIC	99	65	Breakout Debugger Any language New	125	99	Statistician Alpha Computer Service 295 249
Peeks 'n Pokes from MicroHelp	45	39	Essential Comm Library New	185		Strings & Things Alpha Computer Service 70 54
Professional BASIC by Morgan	99	75	Essential Graphics by Essential Software	250 90		Vector87 by Vectorplex Data Systems 150 135
Stay-Res by MicroHelp New	50 95	42 85	Flash-up Windows by Software Bottling GraphiC Mono v2.2 by Sci Endeavors	280	79 209	
True Basic w BASICA Converter New Version	200	99	GraphiC Color v3.0 by Sci Endeavors	350		gss products
True Basic w Converter & Run-time	295	199	GRAFLIB by The Librarian		CALL	GSS Graphics Development Toolkit 495 375
Advanced String Library	50	45	Greenleaf Comm Library by Greenleaf	185		
Asynch Communication Support	50	45	Greenleaf Data Windows by Greenleaf	225		GSS Kernel System for DOS
BASICA Converter	50	45	with Source Code	450		GSS Metafile Interpreter
Btrieve Interface	50	45	Greenleaf Functions by Greenleaf	185		GSS Plotting System 495 375
Developer's Toolkit	50	45	The HAMMER by OES Systems	195		
Formlib	50	45	HALO by Media Cybernetics	300		lattice products
Hercules Graphic Support	50 150	45 109	HELP / Control by MDS	125 185		Lattice C Compiler from Lattice 500 275
Sorting & Searching	50	45	MetaFONTS	80	58	with Library Source Code 900 495
	00		MetaWINDOWS/Plus by Metagraphics	235		C Cross Reference Generator 50 38
blaise products			MetaFONTS/Plus	235		with Source Code 200 145
ASYNCH MANAGER Specify C or Pascal	175	135	On-line Help from Opt-Tech Data Proc	149	109	C-Food Smorgasbord Function Library 150 95
C TOOLS PLUS	175	135	PANEL by Roundhill Computer Systems	295		with Source Code 300 185
EXEC Program Chainer	95	75	PC Lint by Gimpel Software	139		C-Sprite Source Level Debugger 175 129
PASCAL TOOLS	125	99	PLOTHI by The Librarian		CALL	Curses Screen Manager
DACCAL TOOLS & DACCAL TOOLS	100	79 135	PLOTHP by The Librarian	175 175		with Source Code         250         178           dBC dBase File Manager for C         250         178
PASLAL HUILS & PASIAL HILLS	175			150		
PASCAL TOOLS & PASCAL TOOLS 2 RUNOFF Text Formatter	175 50		Vectoro / DV Vectorniex Data Systems New			With Source code
RUNOFF Text Formatter		45 83	Vector87 by Vectorplex Data Systems New Vitamin C by Creative Prog New Version			with Source Code         500         356           LMK Make Facility         195         139
RUNOFF Text FormatterTURBO ASYNCH PLUSTURBO POWER TOOLS PLUS	50 100 100	45 83 83	Vitamin C by Creative Prog New Version VC Screen Forms Designer	225 100		LMK Make Facility 195 139
RUNOFF Text Formatter. TURBO ASYNCH PLUS TURBO POWER TOOLS PLUS VIEW MANAGER Specify C or Pascal	50 100	45 83	Vitamin C by Creative Prog New Version VC Screen Forms Designer Zview by Data Management Consultants	225	CALL 84	LMK Make Facility.       195       139         RPG II Compiler No Royalties       750       635         RPG II Combo with SEU & Sorti Merge       New 1100       939
RUNOFF Text FormatterTURBO ASYNCH PLUSTURBO POWER TOOLS PLUS	50 100 100	45 83 83	Vitamin C by Creative Prog New Version VC Screen Forms Designer Zview by Data Management Consultants	225 100	CALL 84	LMK Make Facility.       195       139         RPG II Compiler No Royalties       750       635         RPG II Combo with SEU & Sortt Merge       New 1100       939         SecretDisk File Encryption Utility.       120       89
RUNOFF Text Formatter. TURBO ASYNCH PLUS TURBO POWER TOOLS PLUS VIEW MANAGER Specify C or Pascal borland products REFLEX Data Base System.	50 100 100	45 83 83	Vitamin C by Creative Prog New Version VC Screen Forms Designer Zview by Data Management Consultants Cobol language Micro Focus COBOL Workhench	225 100 245	CALL 84 189	LMK Make Facility       195       139         RPG II Compiler No Royalties       750       635         RPG II Combo with SEU & Sort1 Merge       New 1100       939         SecretDisk File Encryption Utility       120       89         SideTalk Resident Communications       120       89
RUNOFF Text Formatter. TURBO ASYNCH PLUS TURBO POWER TOOLS PLUS VIEW MANAGER Specify C or Pascal  DOPIAND PRODUCTS REFLEX Data Base System. REFLEX Workshop.	50 100 100 275 150 70	45 83 83 199	Vitamin C by Creative Prog New Version VC Screen Forms Designer Zview by Data Management Consultants Cobol language Micro Focus COBOL Workbench Micro Focus Level II COBOL	225 100 245	CALL 84 189	LMK Make Facility.       195       139         RPG II Compiler No Royalties       750       635         RPG II Combo with SEU & Sort1Merge       New 1100       939         SecretDisk File Encryption Utility.       120       89         SideTalk Resident Communications       120       89         Text Management Utilities       120       89
RUNOFF Text Formatter.  TURBO ASYNCH PLUS  TURBO POWER TOOLS PLUS  VIEW MANAGER Specify C or Pascal  borland products  REFLEX Data Base System.  REFLEX Workshop  REFLEX & REFLEX Workshop.	50 100 100 275 150 70 200	45 83 83 199 99 48 129	Vitamin C by Creative Prog New Version VC Screen Forms Designer	225 100 245	84 189 3379 549	LMK Make Facility.       195       139         RPG II Compiler No Royalties       750       635         RPG II Combo with SEU & Sort! Merge       New 1100       939         SecretDisk File Encryption Utility.       120       89         SideTalk Resident Communications       120       89         Text Management Utilities       120       89         TopView Toolbasket Function Library       250       178
RUNOFF Text Formatter. TURBO ASYNCH PLUS TURBO POWER TOOLS PLUS VIEW MANAGER Specify C or Pascal  DOPIAND PRODUCTS REFLEX Data Base System. REFLEX Workshop.	50 100 100 275 150 70	45 83 83 199	Vitamin C by Creative Prog New Version VC Screen Forms Designer Zview by Data Management Consultants Cobol language Micro Focus COBOL Workhench	225 100 245 4000 1500	3379 549 199	LMK Make Facility.       195       139         RPG II Compiler No Royalties       750       635         RPG II Combo with SEU & Sort1Merge       New 1100       939         SecretDisk File Encryption Utility.       120       89         SideTalk Resident Communications       120       89         Text Management Utilities       120       89

		and the control
logitech products		
LOGIMOUSE C7 Mouse Hardware		85 99
with PLUS Pkg	. 169	135
with PLUS Pkg & CAD Software	. 189 . 219	159 179
with PLUS Pkg & CAD & Paint	w 199	159
MODULA-2/86 Compiler	. 89 . 129	63 103
MODULA-2/86 with PLUS Pkg	. 129	
Library Sources	. 99	
Make Utility	. 29 . 199	
Run Time Debugger	. 69	
Utilities Package	. 49 . 49	
Window Package	. 49	
이 지어 집에 가게 되었다면 나를 보었다면 하다 하게 되었습니다. 하는데 이 사고 있는데 이 사고 있다.	. 89	79
microport product System V/AT by Microport Systems		395
Runtime System (Operating System)	. 159	145
Software Development System		155 155
User Upgrade 3 to Unlimited Users	. 169	
microsoft product	S	
Microsoft BASIC for XENIX		239
Microsoft C with CodeView	. 450 . 700	275 439
for XENIX	. 995	635
Microsoft COBOL Tools with Debugger for XENIX	. 350 . 450	204 309
Microsoft FORTRAN Compiler	. 350	204
Microsoft Learning DOS	. 695 . 50	439 36
Microsoft LISP Common LISP	. 250	169
Microsoft MACH 10 w Mouse & Windows Microsoft MACH 10 Board only	. 549 . 399	385 285
Microsoft Macro Assembler	. 150	98
Microsoft Mouse Bus Version	. 175 . 195	125 135
Microsoft muMath Includes muSIMP	. 300	185
Microsoft Pascal Compiler	. 300 . 695	185 439
Microsoft QuickBASIC	. 99	65
Microsoft Sort	. 99	135 65
Microsoft Windows Development Kit	. 500	309
other languages	. 60	55
CCS MUMPS Single-User by MGlobal	450	379
Janus/ADA C Pack by R&R Software Janus/ADA D Pack by R&R Software	. 95	89 795
Methods Smalltalk by Digitalk	. 79	68
Personal REXX by Mansfield Software Smalltalk/V by Digitalk	125	109
Smalltalk/Comm	. 49	45
SNOBOL4 + by Catspaw	. 95	84
Other products	CALL	CALL
Compact Source Print by Aldebaran Ne Dan Bricklin's Demo Pgm Software Garden	. 75	59
FANSI-CONSOLE by Hersey Micro Ne FASTBACK by 5th Generation Systems		65 149
Informix for DOS by Informix	795	639
InformixAGL for DOS by Informix	. 995 . 795	799 639
Informix SQL for DOS by Informix	90	79
Interactive EASYFLOW by Haventree MKS Toolkit with vi by MKS	. 150 . 139	129 119
Norton Commander by Peter Norton	. 75	55
OPT-Tech Sort by Opt-Tech Data Proc PrintQ by Software Directions	. 149	115 84
Quilt Computing Combo Package	199	169
QMake Program Rebuild Utility SRMS Software Revision Mgmt Sys	. 99 . 125	84 109
screenplay all varieties by Flexus	CALL	CALL
SoftScreen/HELP by Dialectic Systems . Ne Source Print by Aldebaran Labs	97	149 CALL
Taskview by Sunny Hill Software	. 80	65
TLIB by Burton Systems Software Ne Tree Diagrammer by Aldebaran Labs Ne	w CALL	CALL
VTEK 'Term Emulator by Sci Endeavors .	. 150	129
phoenix products	105	115
Pasm86 Macro Assembler Version 2.0 Pdisk Hard Disk & Backup Utility	. 195	115 125
Pfantasy Pac Phoenix Combo Pfinish Performance Analyzer		869 229
Pfix-86 Plus Symbolic Debugger	. 395	229
PforCe Comprehensive C Library		229

#### LOWEST PRICES

Since this ad is prepared in advance of publication, some of our current prices may be lower than what's advertised here. Call for latest pricing.

#### **FREE SHIPPING**

Orders within the USA (including Alaska & Hawaii) are shipped FREE via UPS. Express shipping is available at the shipping carrier's standard rate with no rush fees or handling charges. To avoid delays when ordering by mail, please call first to determine the exact cost of express shipping.

#### CREDIT CARDS

VISA and MasterCard are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include credit card expiration date and telephone number.

#### CODs AND POS

CODs and Purchase Orders are accepted at no extra cost. POs with net 30-day terms are available to qualified US accounts only.

#### **FOREIGN ORDERS**

Shipping charges for foreign and Canadian orders are based on the shipping carrier's standard rate. Since rates vary between carriers, please call or write for the exact cost. Foreign orders (except Canada), please include an additional \$10 for customs form preparation. All payments must be made with US funds drawn on a US bank. Please include your telephone number when ordering by mail. Due to government regulations, we cannot ship to all countries.

#### **VOLUME ORDERS**

Call for special pricing.

#### SOUND ADVICE

Our knowledgeable technical staff can assist in comparing products, answer technical questions and send you detailed product information tailored to your needs.

#### **30-DAY GUARANTEE**

Most of our products come with a 30-day documentation evaluation period or 30-day return guarantee. Please note that some manuafacturers restrict us from offering guarantees on their products. Call for more information.

#### **CALL TOLL FREE**

US	800-336-1166
CANADA	800-225-1166
OHIO	216-877-3781
CUSTOMER SERVICE	216-877-1110

Hours: Weekdays 8:30 AM to 8:00 PM EST. Ohio customers please add 5% state sales tax. Call / write for FREE comprehensive price guide Prices are subject to change without notice.

Plink-86 Plus Overlay Linker	495 125 195 295 195	319 85 115 155 115	
PolyBoost The Software Accelerator Polytron C Beautifier Polytron C Library I Polytron PowerCom Communications PolyLibrarian Library Manager PolyLibrarian II Library Manager PolyMake UNIX-like Make Facility PolyWindows Products All Varieties PolyXREF Complete Cross Ref Utility PolyXREF Complete Cross Ref Utility PolyXREF One language only PVCS Version Control System  Softcraft products	80 49 99 179 99 149 99 CALL 219 129 395	69 45 78 139 78 115 78 CALL 179 109 325	
Btrieve ISAM Mgr with No Royalties	245 245 145 595 595 345	194 194 114 464 464 274	
Brief from Solution Systems . Epsilon Emacs-like editor by Lugaru . KEDIT by Mansfield Software . PC/VI by Custom Software Systems . SPF/PC by Command Technology Corp . Vedit by CompuView . Vedit Plus by CompuView .	195 195 125 149 195 150 185	CALL 159 105 129 139 109 139	
ALICE Interpreter by Software Channels. Btrieve ISAM File Mgr. See SoftCraft FirsTime for Turbo by Spruce Tech. Flash-up Windows by Software Bottling HELP/Control by MDS. On-line Help from Opt-Tech Data Proc. Report Builder by Royal American. New Screen Sculptor by Software Bottling. System Builder by Royal American. New TDebugPLUS by TurboPower Software. Turbo EXTENDER by TurboPower Software. Turbo EXTENDER by TurboPower Software. TurboPower Utilities by TurboPower. TurboPower Utilities by TurboPower. TurboRef by Gracon Services TurboRith Visual Age Debugger TurboWINDOW by MetaGraphics.	95 245 75 90 125 149 75 125 100 60 85 70 99 95 58 80	68 194 59 79 109 109 CALL 94 CALL 49 68 49 85 84 45 65	
Operating System Toolbox  CUNIX Operating system  PCVMS Similar to VAX VMS  XTC Text editor with Pascal source	99 99 99	84 84 79 79	
XENIX System V  XENIX System V Complete System by SCO.  XENIX Development System XENIX Operating Sys Specify XT/AT. XENIX Text Processing Package.	1295 595 595 195	999 499 499 149	
Renix products  Btrieve ISAM File Mgr by SoftCraft . C-ISAM by Informix . c-tree ISAM Mgr w Source by FairCom . dbVISTA Single or Multi User by Raima . dBx with Library Source by Desktop Al . DOSIX User Version by Data Basics . New DOSIX Console Version by Data Basics . New Informix by Informix . Informix 46L by Informix . InformixSQL by Informix . Wicro Focus Level II Compact COBOL Forms-2 . Level II ANIMATOR.	595 319 395 CALL 550 199 399 995 1500 995 595 1000 400 600	464 285 329 CALL 499 CALL 795 1249 795 449 795 319 479	
Microsoft Languages  See Microsoft Section  Networks for XENIX by SCO.  PANEL Screen Designer by Roundhill  REAL-TOOLS Binary Version by PCT New Library Source Version New Complete Source Version New RM / COBOL by Ryan-McFarland  RM / FORTRAN by Ryan-McFarland  SCO Professional Lotus clone by SCO	CALL 595 625 149 399 499 1250 750 795	CALL 495 549 CALL CALL CALL 949 549 595	

### The Peak of Performance



#### SCALE THE HEIGHTS OF PRODUCTIVITY

Sure, you've proven that in your hands a computer is a productive tool. But if you haven't teamed up with a SemiDisk you have heights yet to climb!

#### IT'S NO MERE RAMDISK

SemiDisk has been leading the way for Disk Emulators since their inception. If you've seen RAMdisks you know what it's like to load programs in an

#### **SEMIDISK**

SemiDisk Systems, Inc. P.O. Box GG, Beaverton, Oregon 97075 503-626-3104 instant, and read or write files without delay. Unlike alternatives, the SemiDisk offers up to 8 megabytes of instant-access storage while leaving your computer's main memory free for what it does best - computing!

#### KEEP A GRIP ON DATA

Go ahead, turn off your computer. Take a vacation. With the battery backup option, your valuable data will be there in the morning even if you aren't. You'll sleep better knowing not even a 5 hour blackout will sabotage your files.

#### NEW LOWER SEMIDISK PRICES THAT WON'T SNOW YOU UNDER

SNOW TOO UNL	)ER	
	512K	2Mbyte
IBM PC, XT, AT	\$495	\$995
Epson QX-10	\$595	\$995
S-100, SemiDisk II	\$799	\$1295
S-100, SemiDisk I	\$595	
TRS-80 II, 12, 16	\$695	\$1295
Battery		
Backup Unit	\$130	\$130
		Section 2010 Control of the Control

Software drivers available for CP/M 80, MS-DOS, ZDOS, TurboDOS, and VALDOCS 2.

#### MAC AND AMIGA

#### Listing Three (Listing continued, text begins on page 40.)

```
CreatePort
                                                            ; no preference for signal bit
278
                        move
                                     #-1.DO
                                      AbsExecBase, A6
279
280
                        move.1
                                    AllocSignal
                                                             ;allocate a signal bit for port
                        callsys
                                                             : save signal bit
281
                        move.b
                                    D0, D7
                                    D1
282
                        clr.l
283
                        bset.1
                                     #16,D1
                                                             ; (clear)
                                                             ; (public) requirements; number of bytes needed
284
                        bset.1
                                     #0,D1
                                     #MP STZE. DO
285
                        move
                                      AbsExecBase, A6
                        move.1
286
                                    AllocMem
                                                             ; memory for message port structure
287
                        callsys
                                                             ; save pointer to message port
288
                        move.1
                                     DO, (A3)
                                     DO. A4
289
                        move. 1
                                     #0,D0
290
                        move.1
                                      AbsExecBase, A6
291
                        move.1
                                     FindTask ; initialize task control block
292
                        callsys
                                     A5, LN NAME (A4)
293
                        move.1
                                     #0, LN PRI (A4)
                                                             ;port's priority
294
                        move.b
295
                        move.b
                                     #NT MSGPORT, IN TYPE (A0)
#PA SIGNAL, MP FLAGS (A0)
                                                                   ;type of port
296
                        move b
                                     D7, MP_SIGBIT (A4)
D0, MP_SIGTASK (A4)
                                                             ; signal bit
297
                        move.b
                                                             ; address of task ctrl block
298
                        move.1
                                                             ; is name specified? ; head of list of ports
                                    #0,A5
Port2
299
                         cmp.1
300
                        beg
                        move.1
                                     A4, A1
301
302
                        move.1
                                      AbsExecBase, A6
303
                         callsys
                                     AddPort
                                                             ; add this port to list
304
                         rts
 305
            Port2
                                     MP MSGLIST (A4), A0
 306
                         NEWLIST
                                     AO
                                                             :initialize a new list of ports
307
                         rts
                       Create a standard IO request structure -
;NOTE - this subroutine is an assembly language version of the C source ;code provided in the Amiga ROM kernel manual. It needs error trapping
 after the system calls to be complete.
```

; Load pointer to message port in A3. ; Load address of pointer for standard IO structure in A5.

CreateStdIO			
308	clr.1	D1	
309	bset.1	#16,D1	
310	bset.1	#0,D1	
311	move.1	#IOSTD SIZE, DO	
312	move.l	AbsExecBase, A6	
313	callsys	AllocMem	; space for IO request block
314	move.1	DO, (A5)	;save the pointer
315	move.l	DO, AO	
316	move.b	#NT MESSAGE, LN TYP	PE(A0) ; type of structure
317	move.b	#0, IN PRI (A0)	;priority
318	move.1	A3, MN REPLYPORT (AC	));address of message port
319	rts		

- Subroutine to open the console device -;NOTE - this subroutine is an assembly language version of the C source ; code provided in the Amiga ROM kernel manual. ; Load pointer to WriteMsg in A3. ; Load pointer to ReadMsg in A5.

	OpenConsole		
320	move.1		window record
321	move	#nw SIZE, IO LENGTH (A3) ; si	ze of window record
322	lea	ConDev, AO ; name of device	
323	move.1	#0,D0	
324	move.1	A3, A1	
325	move.1	#0,D1	
326	move.l	AbsExecBase, A6	
327	callsys	OpenDevice	
328	move.1	IO DEVICE (A3), IO DEVICE (A5) ; sav	ve device pointers
329	move.1	IO DEVICE (A3) , DevAdd	
330	move.1	IO_UNIT(A3), IO_UNIT(A5)	
331	rts		

Subroutine to queue up a read request to the console -;NOTE - this subroutine is an assembly language version of the C source ;code provided in the Amiga ROM kernel manual. ; Load pointer to read message in Al. ; Load pointer to storage space for character in A4.

	QueueRead	
332	move	#CMD READ, IO COMMAND (A1) ; type of operation
333	move.1	A4, IO DATA (A1) ; where data should be placed
334	move.1	#1, IO LENGTH(A1) ; number of bytes to read
335	move.1	AbsExecBase, A6
336	callsys	SendIO SendIO
337	rts	(continued on next page)

#### C & PASCAL **PROGRAMMERS**

Blaise Computing provides a broad range of programming tools for Pascal and C programmers, with libraries designed for serious software development. You get carefully crafted code that can be easily modified to grow with your changing needs. Our packages are shipped complete with comprehensive manuals, sample programs and source code.

#### C TOOLS PLUS

\$175.00

NEW! Full spectrum of general-purpose utility functions; windows that can be stacked, removed, and accept user input; interrupt service routines for resident applications; screen handling including EGA 43-line text mode support and direct screen access; string functions; and DOS file handling.

#### PASCAL TOOLS/TOOLS 2

\$175.00

Expanded string and screen handling; graphics routines; easy creation of program interfaces; memory management; general program control; and DOS file support.

#### VIEW MANAGER

\$275.00

Complete screen management; paint data entry screens; screens can be managed by your application program; block mode data entry or fieldby-field control. Specify C or IBM/MS-Pascal.

#### ASYNCH MANAGER

\$175.00

Full featured asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/ XOFF protocol; baud rates up to 9600; modem control and XMODEM file transfer. Specify C or IBM/MS-Pascal.

#### Turbo POWER TOOLS PLUS

\$99.95

NEW! Expanded string support; extended screen and window management including EGA support; pop-up menus; memory management; execute any program from within Turbo Pascal; interrupt service routine support allowing you to write memory resident programs; schedulable intervention code.

#### Turbo ASYNCH PLUS

\$99.95

Complete asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/ XOFF protocol; and baud rates up to 9600.

#### RUNOFF

\$49.95

NEW! Text formatter written especially for programmers; flexible printer control; user-defined variables; index generation; and general macro facility. Crafted in Turbo Pascal.

#### **EXEC**

\$95.00

Program chaining executive. Chain one program from another even if the programs are in different languages. Shared data areas can be specified.

#### **ORDER TOLL-FREE 800-227-8087!**



2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

; Load pointer to Window in A4.

## Is Your Program HALF-FAST?

Speed it up with



# An execution profiler for IBM PCs and compatibles.

Most programs run at less than half the speed that they could. You can optimize almost any code, but where do you start?

A typical program spends 90% of its time in 10% of its code. **The Watcher** identifies that critical 10% for you, so you don't waste your effort on the wrong 90%.

The Watcher is easy to learn and easy to use, and we provide full technical support. Watcher users have increased program performance by as much as 300%. You can get similar results.

The Watcher works with any non-interpretive language on DOS version 2 or 3.

**Turbo Pascal users:** A special section of our manual is dedicated to you!



Plus \$3.00 Shipping & Handling.

To order or for free information, call or write:

Story Brook INC. SOFTWARE

Forest Road Wilton, New Hampshire 03086 (603) 654-2525

Circle no. 287 on reader service card.

#### MAC AND AMIGA

#### Listing Three (Listing continued, text begins on page 40.)

;----- Subroutine to print a single character in a console window ----;NOTE - this subroutine is an assembly language version of the C source; code provided in the Amiga ROM kernel manual.;Load pointer to write message in Al;Load pointer to character to be printed in A4.

	Compart Chara	
338	ConPutChar	#CMD WRITE, IO COMMAND (A1) ; type of operation
339	move.1	A4, IO DATA (A1) ; where data will come from
340	move.1	#1, IO LENGTH (A1) ; number of bytes to output
341	move.1	AbsExecBase, A6
342 343	callsys rts	DoIO
		Data Structures ************************************
344 345	DevAdd TheScreen ds.b	ds.l 1 ns SIZEOF
346	TheWindow ds.b	nw SIZE
347	IntBase	ds.1 1
348	IntName	dc.b 'intuition.library',0
349 350	WindowPtr ds.l WindowTitle	dc.b 'Text Window',0
351	ScreenPtr ds.1	1
352	ScreenTitle	dc.b 'Dr. Dobbs Journal',0
353	ConDev	dc.b 'console.device',0,0
	; NOTE - extra 0 in	cluded to keep addresses even
354	ProjMenu ds.b	mu SIZEOF
355	ProjName dc.b	'Project',0
356	Proj1	dc.b 'New',0
357	ProjText1 ds.b	it_SIZEOF
358	ProjIteml ds.b	mi_SIZEOF
359	Proj2	dc.b 'Open',0,0
360	ProjText2 ds.b	it_SIZEOF
361	ProjItem2 ds.b	mi_SIZEOF
362	Proj3	dc.b 'Save',0,0
363	ProjText3 ds.b	it SIZEOF
364	ProjItem3 ds.b	mi_SIZEOF
265	D44	de h
365 366	Proj4 ProjText4 ds.b	dc.b 'Save As',0 it SIZEOF
367	ProjItem4 ds.b	mi_SIZEOF
368	Proj5	dc.b 'Print',0
369 370	ProjText5 ds.b ProjItem5 ds.b	it_SIZEOF mi_SIZEOF
371	Proj6	dc.b 'Print As',0,0
372 373	ProjText6 ds.b	it SIZEOF
373	ProjItem6 ds.b	mi_SIZEOF
374	Proj7	dc.b 'Quit',0,0
375	ProjText7 ds.b	it SIZEOF
376	ProjItem7 ds.b	mi_SIZEOF
377	EditMenu ds.b	mu SIZEOF
378	EditName dc.b	'Edit',0,0
379	Edit1	dc.b 'Undo',0,0
380 381	EditText1 ds.b EditItem1 ds.b	it_SIZEOF mi_SIZEOF
382	Edit2	dc.b 'Cut',0
383 384	EditText2 ds.b EditItem2 ds.b	it SIZEOF mi_SIZEOF
304	Edititeliz us.b	III_SIZEOF
385	Edit3	dc.b 'Copy', 0, 0
386	EditText3 ds.b	it_SIZEOF
387	EditItem3 ds.b	mi_SIZEOF
388	Edit4	dc.b 'Paste',0
389	EditText4 ds.b	it_SIZEOF
390	EditItem4 ds.b	mi_SIZEOF
391	Edit5	dc.b 'Erase',0
392	EditText5 ds.b	it SIZEOF
393	EditItem5 ds.b	mi_SIZEOF
394	ReadPort ds.1	1
395	ReadMsg	ds.1 1
396 397	ReadName dc.b WritePort ds.l	'Read', 0, 0
398	WriteMsg ds.1	1
399	IntSigBit ds.b	1
400	ConSigBit ds.b	1
401	letter	ds.b 1

**End Listings** 



#### **POWER TOOLS for** SYSTEM BUILDERS™

Ask for Operator 2053

800-543-6277

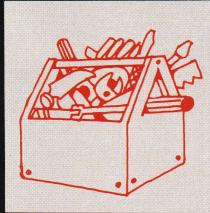
California: 800-368-7600

TSF is owned and operated by programmers, so we understand your needs. We believe and practiice integrity in all business dealings. We advertise real prices and offer our best price to all customers. We provide prompt delivery of current product versions.

We accept checks, Visa, MasterCard and American Express. We charge your card only when we ship and do not add a surcharge. Free UPS delivery on orders over \$100. We allow return privileges on most products. Give us a try!

#### TSF

649 Mission Street San Francisco, CA 94105 (415) 957-0111



Every craftsman has a box filled with his or her personal collection of tools. Knowing which tool to use and how to use it can mean the difference between a quality product and a disaster. TSF's power tools help reduce the drudgery of programming so you can be creative. We suggest the following additions to your toolbox ...

#### Basic: Mac 2 from MicroHelp

expands Basic's horizons with window management, input editing, array sorting and print formatting. It also provides MSDOS/BIOS function calls and allows you to break the 64K data barrier. Written in assembler for high performance. For Bascom, Quick Basic, BASICA and GWBASIC. (List \$75) Only \$65 from TSF. 30 day money back guarantee.

C: Gimpel's PC-Lint evaluates your source code to locate insiduous C bugs like "=" instead of "==" in boolean expressions and mismatched function parameters. Evaluates multiple source files to validate use of all public functions and variables. Compatible with all popular compilers. 30 day trial period (15% re-stock fee). (List \$139) Only \$103 from TSF.

#### All Languages: Periscope II

speeds debugging with multiple windows, symbolic memory addressing, and sophisticated trace and breakpoints. Includes an NMI break switch so you can interrupt programs at any point without pre-setting break points - ends guesswork for hard-tolocate crashes. (List \$129) Only \$108 from TSF. 30 day money back guarantee!

#### Turbo: Kydor's Symbolic

**Debugger** provides symbolic debugging for Turbo Pascal programs. Includes trace, breakpoint, memory display and memory modification. Runs from within Turbo's environment, so you don't loose any Turbo features. Written in assembler for speedy operation and low memory requirements. (List \$49) Only \$40 from TSF.

#### Communications: RamNet from

System Design provides background file transfers and e-mail for your PC. Incoming and outgoing calls are processed automatically while you continue with your normal work. Outgoing calls can be scheduled to automatically take advantage of late night phone rates and to concentrate messages in a multi-node RamNet network. Only \$80 from TSF. A two node starter kit is only \$140. 30 day money back guarantee!

#### Winter Specials

#### **Basic Language**

\$65
160
\$60
\$75
\$80

#### C Language

Datalight C (List \$99)	\$75	
Microsoft C w/Codeview (\$450)	\$285	
Lattice C (List \$500)	\$289	
Mark Williams C (List \$500)	\$319	
Rational Instant C	\$375	
Gimple C-Terp (List \$300)	\$224	
Run/C Professional (List \$250)	\$160	
Run/C Interpreter (List \$120)	\$85	
Gimpel PC Lint (List \$139)	\$103	
Blaise View Manager (List \$275)	\$197	
Lattice Curses (List \$125)	\$90	
Essentails Graphics (List \$250)	\$200	
Blaise C Tools+ (List \$175)	\$125	
Essentials Utility (List \$185)	\$130	
Greenleaf Functions (List \$185)	\$130	
Greenleaf Comm (List \$185)	\$130	
Blaise Async Manager (\$175)	\$125	

... and many more

#### Pascal Language

Microsoft Pascal (List \$300)	\$219
Blaise View Manger (List \$275)	\$197
Blaise Async Manager (\$175)	\$125
Blaise Pascal Tools+ (List \$125)	\$125

#### **Turbo Pascal**

Turbo Pascal BCD & 8087 (\$100)	\$80
Software Channels Alice (\$95)	\$85
Kydor Symbolic Debugger (\$49)	\$40
Blaise Turbo Async (\$100)	\$80
Blaise Power Tools (\$100)	\$80
Borland Turbo Editor (List \$70)	\$50

#### **Support Tools**

Periscope I (List \$295)	\$225
Periscope II (List \$145)	\$108
Phoenix Plink+ (List \$495)	\$325
Phoenix Pfinish (List \$395)	\$275
Phoenix Pfix+ (List \$395)	\$250
Polytron Make (Llst \$99)	\$80
Quilt SRMS + Qmake (List \$199)	\$165
Seidl SVM + SMK (List \$379)	\$330

... call for price list

#### SAPIENS STAR SAPPHIRE

#### Common LISP with Class



#### A common LISP Compiler for the IBM PC TM

- ★ 710 functions
- ★ C source code for Libraries
- ★ Lexical and dynamic scoping
- \* Compiles to C
- ★ Unlimited multi-dimensional arrays
- ★ Object-oriented programming (flavors)
- ★ Numerical functions (bignums up to 128 digits)
- ★ 64-bit virtual memory architecture
- ★ 8 megabytes virtual memory workspace
- ★ LISP to C translator



The Compiler is designed for use in developing programs directly on the PC and for porting applications written on larger machines down to the PC market.

#### \*

#### SYSTEM REQUIREMENTS:

The system requires a DOS-based C compiler which supports huge model. It needs 640K RAM and a hard disk. Programs developed with *Star Sapphire* will run on a system with 256K of RAM. A hard disk is recommended for large applications. The virtual memory manager uses 16-128 kilobytes of RAM at the programmer's discretion.

\* \* \* \$495.00 \* \* \*

Sapiens Software Corporation 236 Mora St. Santa Cruz, CA 95060 408/458-1990

Circle no. 168 on reader service card.

#### 32000 CROSS ASSEMBLER

#### Listing One (Text in December)

```
/* A32000.C - Series 32000 assembler
  850903 rr fix addr ext, scaled index, acp, cxp 0.10
  850902 rr add scaled index logic 0.09
  850828 rr fix enter, setcfg, lpr/spr, index 0.08 850809 rr add equate logic 0.07
   850730 rr add binary search in lookup 0.06
   850729 rr symbol table mods, reglist 0.05
Still need:
   --- register names for lpr/spr
   --- linkable modules
Note: While 68000 is hilo (high-bytes at lower memory
  addresses), 32000 is lohi (low-bytes at lower memory
  addresses, like the Z80).
   This is a 3-pass assembler; 3 passes to make
   sure that relative branches are computed correctly. */
#define EOF -1
                        /* symbol table size */
#define SYMSTZ 1024
char inpbuf[ 256 ]; /* input buffer */
                        /* input counter, pointer */
int inpent, inpptr;
                             /* buff for current word */
char word buffer[ 128 ];
char ambig buffer[ 128 ];
                             /* ambiguous refs here */
char listline[ 81 ];
                        /* line of listing output */
                        /* pointers for list output */
int listop, listop;
                        /* used in gchar() */
int paren = 0;
int brack = 0;
int quote = 0;
                         /* used in inword() */
int iwparen = 0;
                         /* count of errors */
int errors = 0;
                         /* pointer to current word */
char *word;
                         /* filled in by match */
char *ambig[ 10 ];
                         /* count of pointers in ambig[] */
int ambcnt = 0;
                         /* pass = 1, 2 or 3 */
long int asmadr, codadr; /* assembly addr, code addr */
char filename[ 30 ];
                         /* file numbers */
 int fasm, fobj;
 char objbuf[ 64 ];
                         /* object byte buffer */
                         /* addr of first byte of buf */
 long int objadr;
                         /* count of bytes in buffer */
 int objent = 0;
                         /* Symbol table */
 struct {
    char *snam;
                         /* symbol name */
                         /* value */
    long int sval;
 } symbol[ SYMSIZ ];
                         /* count of symbols */
 int syment:
 char hexchr[ 17 ] = "0123456789abcdef";
 /* --- 32000 opcodes --- */
 /* Note: Shortest form of opcode must be listed first. */
 #define MAXOP 149
 /* the opcode binary value should be a string of bits,
    e.g. 0111xxxxx000b the opcode opopt character is used
    to specify special operands, etc. */
 /* opopts used here for the 32000 are:
    blank
              nothing special
         gen short
    b
    C
         gen gen
        00000 short
    d
        gen gen reg
        reglist save/enter
```

```
reglist restore/exit
        00000 gen (sfsr)
        inss/exts
        movs/skps/cmps
        setcfg
        procreg, gen for lpr/spr
        index (operand order)
        ret/rett - postbyte
        movm
        cxp (disp after instruction) */
struct {
   char *onam; /* opcode name */
                 /* operand count, negative if PC-rel */
   int ocnt;
                /* opcode binary value */
   char *obin;
                 /* opcode opopt char */
   char oopt;
} opcode[ MAXOP ] = {
/* Format 1 ops (16) */
   "bsr",
                     "02h",
   "ret",
                               'n',
                     "12h",
   "cxp",
                     "22h",
                1.
                                'p',
   "rxp",
"rett",
                     "32h",
                     "42h",
   "reti",
                     "52h",
   "save",
                     "62h",
   "restore",
                     "72h",
   "enter",
                2,
                     "82h",
   "exit",
                     "92h",
                1.
   "nop".
                     "0a2h"
   "wait",
                     "0b2h",
                     "0c2h",
   "dia",
                0,
                0,
   "flag",
                     "0d2h",
   "svc",
                0,
                     "0e2h",
   "bpt",
                     "Of2h",
/* Conditional branches (15) */
   "beq",
                                 'b',
                      "0ah"
                       "lah",
   "bne",
                -1,
                                 'b',
   "bcs",
                -1,
                      "2ah",
                                'b',
   "bcc",
                -1,
                      "3ah",
                      "4ah",
   "bhi",
   "bls",
                -1.
                      "5ah",
                                 'b'
   "bgt",
                       "6ah",
                                 'b'
   "ble",
                -1,
                      "7ah",
                                 'h'
   "bfs",
                      "8ah",
                                 'b'
                      "9ah",
   "bfc",
                -1,
   "blo",
                      "Oaah",
                                 'b'.
                      "Obah",
   "bhs",
                -1,
   "blt".
                      "Ocah",
                                 'b',
   "bge",
                       "Odah",
                                 'b',
   "br",
                      "Oeah",
                                 'b'
/* Format 2 ops (7) */
   "addg?",
                      "xxxxxxxxx00011iib",
   "cmpq?",
                       "xxxxxxxxxx00111iib",
                                                'e',
   "spr?",
                       "xxxxxxxxxx01011iib",
   "lpr?",
                      "xxxxxxxxx11011iib",
   "seq?",
                      "xxxxx000001111iib",
   "sne?",
                      "xxxxx000101111iib",
                                                'a',
   "scs?",
                      "xxxxx001001111iib",
   "scc?",
                      "xxxxx0011011111ib",
                 1,
                                                'a'
   "shi?",
                      "xxxxx010001111iib",
   "sls?",
                      "xxxxx010101111iib",
   "sgt?",
                      "xxxxx011001111iib",
   "sle?",
                      "xxxxx011101111iib",
   "sfs?",
                      "xxxxx100001111iib",
   "sfc?",
                      "xxxxx100101111iib",
   "slo?",
                      "xxxxx101001111iib",
                                                121
   "shs?",
                      "xxxxx101101111iib",
   "slt?",
                      "xxxxx110001111iib",
                      "xxxxx110101111iib",
                                                'a',
   "sge?",
   "st?",
                      "xxxxx111001111iib",
   "sf?",
                      "xxxxx111101111iib",
/* The acb instruction 3rd operand is a relative jump */
```

(continued on page 85)

# Megamax C

for the

#### Atari ST

"Don't even think about another C compiler" Antic Sept. '86

and Introducing:

# Mac-to-GS C & Mac-to-GS Pascal

Macintosh to Apple IIGS cross compilers. The fastest development systems for the IIGS.

# Megamax Development Systems

Box 851521 • Richardson, TX 75085 (214) 987-4931 • Telex 5106018356

Circle no. 352 on reader service card.

#### **Programmers:**

Turbo-charge Your Productivity with PL/PC

(Programming Language for the PC)

3-5 times more productive than using conventional languages.

Full APL array operators (including matrix inversion and FFT).

Integrated Programming Environment. Modern control and subroutine structure. Local subroutines and variables.

More than 130 built-in mathematical and graphics subroutines.

One conceptual numeric type (including complex numbers).

Large memory model. Source-level debugging.

Source-level debugging Auto-paragraphing.

Built-in full-screen text and data editor. Virtual file variable of up to disk capacity.

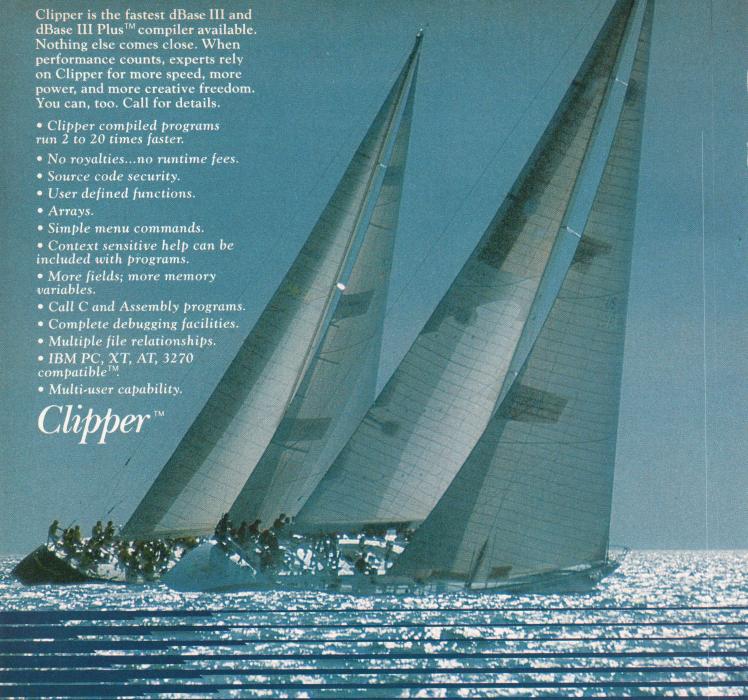
For only \$89 8087 version \$159 Demo version (includes reference manual) \$16

Requirements: IBM PC or compatibles DOS 2.11 or higher 360K bytes memory

#### CREATIVE COMPUTER SOFTWARE

117 York St., Sydney, NSW 2000, Australia. Phone: (02) 261 1611 Fax: (02) 264 7161

Circle no. 348 on reader service card.



CLIPPER. THE dBASE COMPILER. A WINNING PERFORMANCE EVERY TIME.



nantucket™

Nantucket Corporation 5995 South Sepulveda Boulevard Culver City, California 90230 (213) 390-7923 Outside California call toll-free:

1-800-251-8438

dBase, dBase III, and dBase III Plus are trademarks of Ashton-Tate, Inc. IBM PC, XT, AT, and 3270 are trademarks of International Business Machines Corporation.

Clipper and Nantucket are trademarks of Nantucket Corporation.

LOOK FOR CLIPPER™ IT MAKES NETWORKING EASY.

Circle no. 220 on reader service card.

#### 32000 CROSS ASSEMBLER

#### Listing One (Listing continued)

```
'e',
   "acb?"
                 -3
                        "xxxxxxxxx10011iib".
                                                                       "inss?".
                                                                                           "xxxxxxxxxx0010ii11001110b",
   "movq?"
                 2.
                       "xxxxxxxxx10111iib",
                                                'e'.
                                                                       "exts?",
                                                                                           "xxxxxxxxxx0011ii11001110b".
                                                                                                                             111,
                                                                                     4.
                                                                       "movxbw?".
                                                                                           "xxxxxxxxx0100ii11001110b",
                                                                                                                             'c',
/* Format 3 instructions (7) */
                                                                       "movzbw?"
                                                                                                                             'c',
                                                                                           "xxxxxxxxx0101ii11001110b".
                                                                       "movz?d".
                                                                                           "xxxxxxxxxx0110ii11001110b".
                                                                                                                             'c',
   "cxpd"
                       "xxxxx000001111111b".
                                                'a',
                 1.
                                                                       "movx?d",
                                                                                           "xxxxxxxxxx0111ii11001110b".
                                                                                                                             'c',
   "bicpsr?",
                       "xxxxx001011111iib",
                 1.
                                                'a',
                                                                       "mul?",
                                                                                           "xxxxxxxxxx1000ii11001110b",
                                                                                                                             'c'.
   "jump",
                 1,
                       "xxxxx01001111111b".
                                                'a',
                                                                       "mei?",
                                                                                           "xxxxxxxxx1001ii11001110b",
                                                                                                                             'c'
                                                'a',
   "bispsr?",
                 1.
                       "xxxxx011011111iib",
                                                                       "dei?",
                                                                                           "xxxxxxxxx101111111001110b".
                                                                                                                             'c',
   "adjsp?",
                       "xxxxx1010111111ib".
                                                'a',
                                                                       "quo?",
                                                                                           "xxxxxxxxx1100ii11001110b",
                                                                                                                             'c',
                                                                       "rem?",
   "jsr",
                       "xxxxx11001111111b",
                 1.
                                                'a'.
                                                                                                                             'c',
                                                                                           "xxxxxxxxx1101ii11001110b",
   "case?"
                       "xxxxx111011111iib".
                                                'a',
                                                                        "mod?",
                                                                                           "xxxxxxxxxx1110ii11001110b",
                                                                                                                             'C'
                                                                       "div?",
                                                                                           "xxxxxxxxxx1111ii11001110b",
                                                                                                                             'c',
/* Format 11 ops (16) -
   moved here so wildcards won't interfere */
                                                                    /* Format 8 ops (8) */
   "addf"
                       "xxxxxxxxxx00000110111110b",
                                                                       "ext ?".
                                                                                           "xxxxxxxxxxxxx0ii00101110b".
                                                                                                                             'm',
   "addl",
                       "xxxxxxxxx00000010111110b",
                                                        'c',
                                                                       "cvtp",
                                                                                           "xxxxxxxxxxxxx01101101110b",
                                                                                                                             'm',
                                                                                     3.
   "movf",
                       "xxxxxxxxxx00010110111110b",
                                                        'c',
                                                                       "ins?"
                                                                                     4.
                                                                                           "xxxxxxxxxxxxx0ii101011110b".
                                                                                                                             'm'.
   "movl",
                 2,
                       "xxxxxxxxxx00010010111110b",
                                                                       "check?",
                                                         101
                                                                                           "xxxxxxxxxxxxx0ii11101110b",
                                                                                                                             'm',
                                                                                     3,
   "cmpf",
                       "xxxxxxxxxx00100110111110b",
                 2.
                                                         'C'
                                                                       "index?".
                                                                                           "xxxxxxxxxxxxx1ii00101110b".
                                                                                                                             'm'.
   "cmpl",
                       "xxxxxxxxxx00100010111110b",
                                                        'c',
                 2.
                                                                       "ffs?",
                                                                                           "xxxxxxxxxx0001ii01101110b",
                                                                                                                             'C'
   "subf",
                       "xxxxxxxxx01000110111110b",
                 2.
                                                         'c',
                                                                                           "xxxxxxxxxx0011ii101011110b",
                                                                       "movsu?"
                                                                                                                             'C'
   "subl",
                       "xxxxxxxxxx01000010111110b",
                 2.
                                                        'c',
                                                                       "movus?",
                                                                                           "xxxxxxxxxx0111ii10101110b",
                                                                                                                             'c'
   "negf",
                 2,
                       "xxxxxxxxxx010101101111110b",
                                                        'c',
   "negl",
                 2,
                       "xxxxxxxxxx01010010111110b",
                                                        'c'.
                                                                    /* Format 9 ops (12) */
   "divf",
                 2.
                       "xxxxxxxxx10000110111110b",
                                                        'c',
   "divl",
                       "xxxxxxxxx10000010111110b",
                                                        'c',
                                                                       "movlf",
                                                                                           "xxxxxxxxxx0101ii001111110b".
                                                                                                                             'c'.
   "mulf",
                 2.
                       "xxxxxxxxxx11000110111110b".
                                                        'c',
                                                                       "movfl",
                                                                                     2.
                                                                                           "xxxxxxxxxx0111ii00111110b",
                                                                                                                             'c',
   "mull",
                       "xxxxxxxxx110000101111110b",
                                                        'c'.
                                                                       "mov?f".
                                                                                           "xxxxxxxxxx0001ii00111110b",
                                                                                     2.
                                                                                                                            'c',
   "absf",
                       "xxxxxxxxxx110101101111110b",
                                                        'C'
                                                                       "mov?1",
                                                                                     2,
                                                                                           "xxxxxxxxxx0000ii00111110b",
                                                                                                                             'c',
   "absl",
                       "xxxxxxxxx11010010111110b",
                                                        'c'
                                                                       "lfsr",
                                                                                           "xxxxx0000000111100111110b",
                                                                                     1.
                                                                       "sfsr",
                                                                                           "00000xxxxx11011100111110b",
                                                                                                                             'h',
/* Format 4 instructions (12) */
                                                                       "roundf?",
                                                                                           "xxxxxxxxxx1001ii00111110b".
                                                                                                                             'c',
                                                                       "roundl?",
                                                                                           "xxxxxxxxxx1000ii001111110b"
                                                                                                                             'c',
   "add?",
                       "xxxxxxxxxx0000iib",
                                                'c',
                                                                       "truncf?",
                                                                                           "xxxxxxxxxx1011ii00111110b".
                                                                                                                             'c',
                                                'c',
   "cmp?",
                       "xxxxxxxxxx00011ib",
                 2.
                                                                       "truncl?",
                                                                                           "xxxxxxxxxx1010ii00111110b",
                                                                                                                            'c',
   "bic?"
                       "xxxxxxxxxx0010iib"
                                                'c',
                 2.
                                                                       "floorf?".
                                                                                           "xxxxxxxxxx1111ii00111110b",
                                                                                                                            'c',
   "addc?",
                       "xxxxxxxxxx0100iib",
                 2.
                                                'c',
                                                                       "floorl?".
                                                                                           "xxxxxxxxx1110ii00111110b",
                                                                                                                            'c'.
   "mov?",
                       "xxxxxxxxxx0101iib",
                 2,
                                                'c',
   "or?",
                       "xxxxxxxxxxx0110iib",
                 2,
                                                'c',
                                                                    /* Format 14 instructions (4) */
   "sub?"
                 2,
                       "xxxxxxxxxx1000iib",
                                                'c'.
   "addr",
                 2,
                       "xxxxxxxxxx100111b",
                                                'c',
                                                                                     1,
                                                                       "rdval".
                                                                                           "xxxxxxxxx000001100011110b",
                                                                                                                            'a'.
   "lxpd",
                      "xxxxxxxxxx100111b",
                                                'c'.
                                                                       "wrval",
                                                                                     1,
                                                                                           "xxxxxxxxx000011100011110b".
                                                                                                                            'a',
                                               'c',
   "and?"
                       "xxxxxxxxxx1010iib",
                                                                       "lmr",
                                                                                     2,
                                                                                           "xxxxxxxx000101100011110b",
                                                                                                                             101
   "subc?",
                      "xxxxxxxxx1100iib",
                                                'c',
                                                                       "smr",
                                                                                     2.
                                                                                           "xxxxxxxxx000111100011110b".
                                                                                                                            'e'
   "tbit?",
                       "xxxxxxxxxx1101iib",
                                               'c',
                                                                    };
   "xor?",
                       "xxxxxxxxxx1110iib",
                                               'c',
                                                                       /* Address Mode Table */
/* Format 5 instructions (4) */
                                                                    #define MAXAM 42
   "movst"
                       "00000xxx100000ii00001110b",
                                                        111,
                      "00000xxx000000ii00001110b",
                                                        'j',
   "movs?",
                 1,
                                                                    struct {
                                                        · · · · ·
   "cmost".
                 1,
                       "00000xxx100001ii00001110b",
                                                                       char *mstr; /* mode match string */
   "cmps?",
                      "00000xxx000001ii00001110b",
                 1.
                                                                       char *gstr;
                                                                                   /* output string to insert (gen) */
   "skpst",
                       "00000xxx100011ii00001110b",
                                                        111,
                 1,
                                                                                     /* count of ambigs to be put into
                                                                       int ment:
   "skps?",
                       "00000xxx000011ii00001110b",
                 1.
                                                                                    extension bytes */
                                                                       char mont:
                                                                                     /* mode option */
   "setcfg",
                                                                    } admode[ MAXAM ] = {
                      "00000xxxx000101100001110b",
                                                        'k'
/* Format 6 ops (14) */
                                                                    /* Scaled index modes */
   "rot?".
                 2.
                      "xxxxxxxxx0000ii01001110b",
                                                        'c'.
                                                                       "*[r?:b]",
                                                                                     "11100",
                                                                                                      's',
                                                                                                 1,
   "ash?"
                       "xxxxxxxxxx0001ii01001110b",
                                                        'c',
                                                                       "*[r?:w]",
                 2,
                                                                                     "11101",
                                                                                                      's',
                                                                                                 1,
                                                                       "*[r?:d]",
   "cbit?"
                                                        'c',
                      "xxxxxxxxxx0010ii01001110b",
                                                                                     "11110",
                                                                                                 1,
                                                                                                       's',
   "cbiti?",
                                                                       "*[r?:q]",
                      "xxxxxxxxxx0011ii01001110b",
                                                        'c'
                                                                                     "11111",
                                                                                                       's'
   "lsh?",
                      "xxxxxxxxxx0101ii01001110b",
                                                        'c',
   "sbit?"
                 2,
                      "xxxxxxxxxx0110ii01001110b",
                                                        'c',
                                                                   /* Simple register modes */
   "sbiti?",
                 2.
                      "xxxxxxxxxx0111ii01001110b",
                                                        'c'.
   "neg?",
                      "xxxxxxxxx1000ii01001110b",
                                                        'c',
                                                                               "00000",
                 2.
                                                                       "r0"
                                                                                                        /* main registers */
   "not?"
                 2,
                      "xxxxxxxxxx1001ii01001110b",
                                                        'c',
                                                                       "r1",
                                                                               "00001",
                                                                                           0.
   "subp?"
                                                        'c',
                      "xxxxxxxxxx1011ii01001110b".
                                                                       "r2",
                                                                               "00010",
   "abs?",
                      "xxxxxxxxxx1100ii01001110b"
                 2.
                                                        'c',
                                                                       "r3"
                                                                               "00011",
                                                                                           0,
   "com?"
                      "xxxxxxxxxx1101ii01001110b",
                                                        'c',
                                                                       "r4",
                                                                               "00100",
   "ibit?".
                      "xxxxxxxxx1110ii01001110b",
                                                                       "r5",
                                                                               "00101",
                                                        'c'.
                                                                               "00110",
   "addp?",
                      "xxxxxxxxxx1111ii01001110b",
                                                                       "r6",
                                                                                           0.
                                                                       "r7",
                                                                               "00111",
                                                                                           0.
/* Format 7 ops (15) */
                                                                       "f0",
                                                                               "000000".
                                                                                           0.
                                                                                                        /* floating point */
   "movm?",
                      "xxxxxxxxxx0000ii11001110b",
                                                        101.
                                                                       "f1",
                                                                               "00001",
                                                                                           0.
                      "xxxxxxxxxx0001ii11001110b",
                                                                       "f2"
                                                                               "00010",
   "cmom?"
                                                        'c'
                                                                                                        (continued on next page)
```



On-Line HELP! Function Library
HELP! is an on-line utility library that gives your software instant,
context—sensitive, pop—up HELP! windows at the touch of a key. Link your
C code with the HELP! library, tell HELP! which window is current, and HELP! does the rest - fast. HELP! writes directly to video memory with no flicker

HELP! Runs on PCs and compatibles, B+W or CGA. Source code is

**HELP!** Composer

Compose your HELP! windows interactively. Control HELP! window text, size, colors, position, borders, titles. The HELP! Composer runs as a standalone utility or link it along with your programs to watch each HELP! window take form against the backdrop of your own screen designs. The HELP! Composer builds an ASCII text file to describe all the HELP! windows for each application.

Complete Windows and Pop-Down Menu Library

A bonus: Use the window and pop—down menu functions from the HELP! library in your own programs. This is a complete C window package. Open and close windows, fill windows with text, scroll, select with a cursor bar, promote a window from the background, move a window.

Window Text Editor

Use the full-featured, window-oriented HELP! Text Editor to collect text data into your application. Open a window and call the editor.

HELP! links with programs compiled by most MS—DOS C compilers: Aztec, CI—C86, Datalight, DeSmet, Eco—C88, High C, Lattice, Lets C, Microsoft, Whitesmiths, Wizard, Most memory models are supported. The HELP! distribution package includes libraries, a C demo program, and a complete Programmer's Manual.

HELPI: \$49. HELPI with Source Code: \$149. Demo diskette: \$10. deducted from your order. Specify compiler. MasterCard and VISA are accepted. (FL residents add 5% sales tax.)

C SOFTWARE TOOLSET

2983 Newfound Harbor Drive — Merritt Island, FL 32952 — (305) 453—0257

Circle no. 235 on reader service card.

#### program with power!

#### PC/POWER<sup>TM</sup>

APPLICATION DEVELOPMENT & MANAGEMENT INTRODUCTORY OFFER: \$95 Including s & h

Runs on all 100% PC compatibles!

- · Supports applications in a variety of languages through program calls - NOT a code generator
  - -C, PASČAL, BASIC, ASSEMBLER
- Screen painter/editor create language and program independent data entry/display screens
  - Alphanumeric, Integer, Long, Floating point fields
- Build your own POP-UP menu/selection windows under your program
- Supports CGA, EGA, and monochrome display adapters
- Indexing function for cataloging programs and applications provides easy management of ALL of your PC programs and packages.
- -Integrate existing programs into an application Supplies EXEC function to pass control between programs
- Even pass data between programs in different languages · Development utility applications included as samples
- Entire package written in assembler; lightning FAST!

#### COMPLETE DEVELOPMENT & RUNTIME SYSTEM! \$95 → NO ROYALTIES ←\$95

(Mass. residents add 5% sales tax)

ORDERS OR INQUIRIES (800) 628-2828 ext. 712



#### ORDER NOW!



BEACON STREET SOFTWARE, INC. P.O. BOX 216 • BEACON HILL • BOSTON, MA 02133

Circle no. 267 on reader service card.

#### 32000 CROSS ASSEMBLER

#### Listing One (Listing continued)

```
"f3",
           "00011",
  "f4",
"f5",
           "00100",
                       0.
           "00101",
                       0,
   "f6",
           "00110",
                       0,
  "f7",
           "00111",
/* Indexed addressing modes */
   "* (r0) ",
              "01000",
                                       /* indexed */
              "01001",
   "* (r1) ",
                          1,
   "* (r2) ",
              "01010",
                          1,
              "01011",
   "* (r3) ",
   "* (r4) ",
              "01100",
                          1,
              "01101",
  "* (r5) ",
   "* (r6) ",
              "01110",
   "* (r7) ",
   "*(*(fp))",
                 "10000", 2,
                                       /* frame ptr */
   "* (* (sp))",
                "10001",
                                         /* stack mem */
                                  'r',
   "* (* (sb))",
                                         /* static mem */
                "10010",
                                  'r'.
                 "10100",
                                         /* immediate */
   "@*",
                "10101",
                                         /* absolute */
   "ext (*) +*",
                "10110",
                                         /* external */
                "10111",
                                         /* top of stack */
   "tos",
   "*(fp)",
                 "11000",
                                         /* frame mem */
                 "11001",
                                         /* stack mem */
   "* (sp) ",
                                         /* static mem */
   "* (sb) ",
                 "11010".
   ".+*",
                 "11011",
                            1,
                                         /* program mem */
   "[*]",
                             '1',
                                    /* register list */
                       0.
/* catch-all */
          ....
                1.
                      'w' /* fits no pattern */
/*---MAIN PROGRAM---*/
main ( argc, argv )
int argc;
char *argv[];
   int i:
   puts( "\nA32000 v0.10" );
   if( argc < 2 ) {
      puts ( "\n?No file name specified" );
      exit(1);
   syment = 0;
   for ( pass = 1; pass <= 3; ++pass ) {
      makename( argv[ 1 ], ".s" );
      fasm = fopen( filename, "r" );
      if ( fasm == 0 ) {
         puts ( "\n?Unable to open source file" );
          exit(1);
      if ( pass == 3 ) {
         makename( argv[ 1 ], ".hex" );
          fobj = fopen(filename, "w");
          if(! fobj) {
             puts ( "\n?No directory space" );
             exit(1);
      puts( "\nPass " );
      putchar ( pass + '0' );
      asmadr = 0;
      codadr = 0:
      if ( pass == 3 ) {
```

```
obiflush();
         listnl();
      inpload():
      while ( gword () ) {
          if ( match ( word, "end" )) break;
/* Each word is processed by the following nested if
   statement, which attempts to identify what it is.
   Note that any successful identification stops the
   process of the statement. */
          if(! islabel(word))
             if(! ispseudo( word ))
                if(! isopcode( word ))
                   if( ! isequate( word ))
                      error( '?', word );
      fclose ( fasm );
/* Sort symbols after pass 1. */
      if ( pass == 1 ) sortsyms();
      if ( pass == 3 )
          objflush();
          putc( ':', fobj );
                                 /* write eof record */
         for( i = 0; i < 10; ++i) putc('0', fobj);
putc('\n', fobj);
         fclose (fobj);
   }
   listpr();
   puts( "\n\n" );
   dumpsyms();
   if ( errors )
      puts ( "\n---Fix errors and reassemble---" );
/* Construct a filename from two strings. */
makename (p, q)
char *p, *q;
   char *r:
   r = &filename[ 0 ];
   while( *p ) *r++ = *p++;
   while( *q ) *r++ = *q++;
   *r = '\0';
/* Check to see if the word is a label, and if it is, add
   its value to the symbol table */
int islabel ( w )
char *w:
   while( *w ) ++w;
   if( *--w != ':' ) return 0;

*w = '\0'; /* take off the colon */
   addsymbol ( word, codadr );
}
/* Check the word to see if it is a pseudo-op. */
int ispseudo ( w )
char *w:
   long int getarg(), temp;
   if( match( w, "org" )) {
   asmadr = getarg();
   codadr = asmadr;
   if ( pass == 3 ) objflush();
```

(continued on next page)



to

C

#### the dBx™ translator

- dBx produces quality C direct from dBASE II or III programs.
- Move dBASE programs to UNIX or other machines.
- Improve program speed and reliability.
- Support multi-user/network applications.
- With power guidebook of conversion hints.
- Includes full screen handler and uses your current C database manager.
- May be used to move existing programs or help dBASE programmers learn C easily.
- For MSDOS, PCDOS, UNIX, XENIX, Macintosh, AMIGA.
   (Uses ANSI.SYS driver on MSDOS, CURSES under UNIX)
- Priced from \$350, also available from distributors

#### dBx is a trademark of Desktop Ai

1720 Post Road E., Westport, CT 06880 MCIMAIL • DESKTOPAI Phone • 203•255•3400 Telex • 6502972226MCI

Circle no. 258 on reader service card.



## Transform Your Programs with

#### **CPP—C Preprocessor Plus**

#### Includes ALL features of the standard C preprocessor.

- Define arbitrarily complex macros with #define command.
- Include and nest files to any depth with #include command.
- Include lines with #if, #ifdef and #ifndef commands.
  Define multiple constants with #enum command.
- Optional extra feature: Imbed formatting or other commands in your source code. (Lines starting with . or \* are ignored.)

#### Fast and flexible

- 30 times faster than the Preprocessor published in Dr. Dobb's Journal.
- Can be used for any language, including assembler.
- Can be used as a stand-alone macro/include processor.
- Code can be used as the lexical analyzer for parsers or assemblers.

#### Complete

- You get complete SOURCE CODE in standard C.
- You get everything you need to use CPP immediately.
- CPP is unconditionally guaranteed. If for any reason you are not satisfied with CPP, your money will be refunded promptly.

#### Price: \$95.

Enteleki, Inc. 210 N. Bassett St., Room 101 Madison, WI 53703 Tele. (608) 258-7078

TO ORDER: Specify both the operating system (MS-DOS, CP/M 80 or CPM 68K) and the disk format (8 inch CP/M or the exact type of 5½ inch disk). Send a check or money order for \$95 (\$105 for foreign orders). Foreign checks must be denominated in U.S. dollars drawn on a U.S. bank. Sorry, I do NOT accept phone, credit card or COD orders. Please do NOT send purchase orders unless a check is included.

Circle no. 90 on reader service card.

#### 32000 CROSS ASSEMBLER

#### Listing One (Listing continued.)

```
return 1;
  if( match( w, "db" )) {      /* Note: Allow msgs? */
                              /* get argument */
     temp = getarg();
     objout ( temp & 0xFF );
                             /* output byte */
     return 1:
  if ( match ( w, "dw" )) {
     objout(( temp >> 8 ) & 0xFF );
                                     /* output msb */
     return 1;
  if ( match ( w, "dd" )) {
                             /* get argument */
     temp = getarg();
                              /* output lsb */
     objout ( temp & OxFF );
     objout (( temp >> 8 ) & 0xFF );
     objout (( temp >> 16 ) & 0xFF );
     objout(( temp >> 24 ) & 0xFF ); /* output msb */
     return 1;
  if ( match ( w, "even" ) && ( codadr & 1 )) {
     objout (0); /* send 1 byte to go to word bndry */
     return 1:
  return 0;
/* Check to see if the word is an opcode, and if it is,
  get any operands required and generate code. */
int isopcode ( w )
char *w:
   long int value(), bitbin(), decbin(), o, ocodadr;
   char opbuf[ 33 ], bytbuf[ 33 ], extbuf[ 128 ];
  char opopt, modopt, opsiz, opent;
  int i, j, k, l;
char *p, *q, *cpystr(), *regbits();
   /* postbytes & scaled indexes */
   int opexbt[ 4 ], opexct;
   int adexct, adexln[ 8 ];
   char *adexpt[ 8 ], *eoadex; /* addressing extensions */
                       /* save addr of begin of instr */
   ocodadr = codadr:
   opexct = 0;
                       /* no postbytes as yet */
                        /* no extensions as yet */
   adexct = 0:
   eoadex = &extbuf[ 0 ]; /* point to begin of extbuf */
   for ( i = 0; i < MAXOP; ++i )
      if ( match ( w, opcode[ i ].onam )) {
      opopt = opcode[ i ].oopt;
      if ( opopt == 'x' ) {
         error('x', w); /* unimplemented instruction */
         return 1:
      p = cpystr( opcode[ i ].obin, &opbuf[ 0 ] );
/* see if length modifier */
      if ( ambent > 0 ) {
         p = & opbuf[0];
         opsiz = *ambig[ 0 ];
         while( *p && *p != 'i' ) ++p;
         if( ! *p ) error( 'l', w );
         switch ( opsiz ) {
```

```
case 'b' : *p++ = '0';
               *p++ = '0';
              break;
           case 'w' : *p++ = '0';
               *p++ = '1';
              break:
           case 'd' : *p++ = '1';
               *p++ = '1';
        }
/* now parse operands */
/* get count of operands.
   Take abs value (neg = PC-relative) */
      opent = opcode[ i ].ocnt;
     if (opent < 0 ) opent = 0 - opent;
     p = &opbuf[ 0 ]; /* modified parts start at beg. */
      for( j = 0; j < opent; ++j ) {
                      /* get operand */
/* find addr mode */
         k = 0:
         while(( k < MAXAM )
            && ! match ( word, admode[ k ].mstr )) ++k;
         modopt = admode[ k ].mopt;
/* move bit string into place */
         g = admode[ k l.gstr:
/* if opopt h, sfsr, skip 5 bits */
         if ( opopt == 'h' ) p += 5;
/* for most opopts, move the bits in */
         if(( opopt == ' ')
            || ( opopt == 'a'
            || ( opopt == 'c' )
            || ( opopt == 'e' && j == 1 )
            || ( opopt == 'h' )
            || ( opopt == 'i' && j < 2 )
            || ( opopt == 'l' && j == 1 )
            || ( opopt == 'm' && j > 0 && j < 3 )
            || ( opopt == 'o' && j < 2 ))
            while( *q ) *p++ = *q++;
/* Double the effort for scaled index mode. create an
   extension postbyte opexbt[] with basemode as upper
   5 bits, reg as lower 3 bits. */
         if( admode[ k ].mopt == 's' ) {
            1 = ( *ambig[ 1 ] ) & 7;
            q = cpystr( ambig[ 0 ], &bytbuf[ 0 ] );
/* find basemode */
            k = 0;
            while(( k < MAXAM )
               && ! match ( &bytbuf[ 0 ],
               admode[ k ].mstr )) ++k;
            modopt = admode[ k ].mopt;
/* move bit string into postbyte. use bitbin, because
   value() destroys ambig[] array which we still need. */
            q = cpystr(admode[k].gstr,
               &bytbuf[ 0 ] );
                                /* back up to null */
            --q;
           *q++ = '0' + (( 1 >> 2 ) & 1 );
           *q++ = '0' + ((1 >> 1) & 1);
```

```
*q++ = '0' + (1 & 1);
             *q = '\0';
             opexbt[opexct++] = bitbin(&bytbuf[0]);
             g = admode[ k ].gstr:
/* funny handling of reg: index operation (opopt 'm') */
          if( opopt == 'm' && j == 0 ) {
             p = &opbuf[ 10 ]; /* off to reg */
             q += 2;
                                 /* skip 0 bits */
             while( *q ) *p++ = *q++;
             p = &opbuf[ 0 ];
                                /* reset */
/* move ambigs into extension bytes. set length to
   variable (0). For some addressing modes, the
   extensions go in in reverse order */
          if ( modopt == 'r' )
             for( 1 = admode[ k ].mcnt - 1; 1 >= 0;
                --1 ) {
                adexpt[ adexct ] = eoadex;
                adexln[ adexct ] = 0;
                ++adexct:
                eoadex = cpystr(ambig[l], \
                   eoadex );
          } else for( 1 = 0; 1 < admode[ k ].mcnt; ++1 ) {</pre>
                adexpt[ adexct ] = eoadex;
                adexln[ adexct ] = 0;
                ++adexct;
                eoadex = cpystr( ambig[ 1 ], \
                   eoadex );
/* special logic for register list for "enter", "save",
   "restore", "exit" */
          if(( j == 0 && opopt == 'f' ) || opopt == 'g' ) {
            adexpt[ adexct ] = eoadex;
             adexln[ adexct ] = 1;
             ++adexct:
           . eoadex = regbits ( word, eoadex, opopt );
/* shorten extension to 1 byte for enter, return */
         if(( j == 1 && opopt == 'f' ) || opopt == 'n' ) {
   if( adexct > 0 ) adexln[ adexct - 1 ] = 1;
   else error( 'e', w );
/* opopts 'e' or 'd': immed data becomes 4 bit value */
         if(( j == 0 && opopt == 'e' ) || opopt == 'd' ) {
             if(! adexct ) error('e', w);
                else {
                l = value( adexpt[ --adexct ] );
                p = &opbuf[ 5 ];
                *p++ = '0' + (( 1 >> 3 ) & 1 );
                *p++ = '0' + ((1 >> 2 ) & 1 );
                *p++ = '0' + (( 1 >> 1 ) & 1 );
                *p = '0' + (1 & 1);
               p = &opbuf[ 0 ];
            }
/* opopt 'i': combine last two ambigs into one postbyte.
  Put it in bytbuf and set length to 1 */
         if ( opopt == 'i' && j == 3 ) {
            if( adexct < 2 ) error( 'e', w );
            else {
               l = ( value( adexpt[ --adexct ] ) - 1 )
                  & 31:
               1 += ( value( adexpt[ --adexct ] )
                  & 7 ) << 5;
                bytbuf[ 0 ] = '0' +
                   ((1/100) % 10);
                bytbuf[ 1 ] = '0' + ((1 / 10) % 10);
                                    (continued on next page)
```

#### **TURBO PASCAL GENERATOR**

#### THE GTP PROFESSIONAL MODEL

Generate error-free Turbo Pascal source code for:

DATA BASES MENUS SCREENS REPORTS

Build complete, working programs in minutes!

#### **FEATURES**

- Indexed Data Bases
- Multiple Screens
- Automatic Updating
- Built-in Edits
- Retrieval Facility
- Automatic File Build
- Quick Screen Handling
- Speedy DB Access Context Sensitive HELP
- Full Keyboard Support

- 1. Paint-the-screen
- 2. Define fields & calculations 3. Generate compile & RUN

#### FLEXIBLE

- 100% Turbo Pascal
- Modify in Pascal under Turbo's Editor
- Extensions are easy with 130 page
- Programmer Reference manual 100 documented source code
- routines included

NOT Copy Protected NO Royalty Fees NO Run-time Library Required

Requires: IBM PC (100% compatible) 256K RAM 2 Disk Drives PC-DOS 2.0 + Turbo Pascal 3.0 from Borland International

#### **PRICE \$200.00 CALL FOR QUANTITY DISCOUNTS**

VISA/MC — Check — Money Order — No COD or Purchase Orders Texas Residents: add \$12.25 Sales Tax Outside US & Canada; add \$10,00 Air Postage and make payment by credit card or money order in U.S. Funds.



#### **ALLEN, EMERSON & FRANKLIN**

P.O. Box 928 Katy, TX 77492 (713) 391-8570

Circle no. 219 on reader service card

#### Publication Quality Scientific Graphics



Over 100 C routines make scientific plotting easy

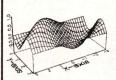
- → linear, log, & polar plots
- → bar charts & Smith charts
- -> contour plots with labels
- → 3-D curves, 3-D surfaces
- → 4 curve types, 8 markers, errorbars
- → 14 fonts, font editor
- → unlimited levels of superscripts
- → 4096 x 3120 resolution in 16 colors on EGA, Tecmar, Sigma boards
- → zoom, pan, window and merge plots
- → high resolution printer dumps

SOURCE INCLUDED for personal use only

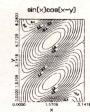
\$350. Demo \$8



256k, IBM, AT&T, Corona PCs, DOS 2.xx, 3.xx Most boards, printers, and plotters supported Microsoft, Lattice, DeSmet, Aztec, C86 compilers







Scientific Endeavors Corporation Route 4, Box 79 Kingston, TN 37763 (615) 376-4146

Circle no. 210 on reader service card.

# BRING YOUR HARD DISK BACK UP TO SPEED WITH H.D. TUNEUP!

The harder you work, the more files you put on your hard disk, the faster DOS works to fragment those files. Fragmented files make you wait longer for file loads and saves. **H.D. Tuneup** rearranges the data on your hard disk for the fastest possible file i/o times. Your disk will speed along like new.

"Much better than Disk Optimizer™" - Philadelphia, PA

Requires IBM PC/XT/AT compatability, DOS 2.x or higher and at least 196K. Hard disks up to 32mb may be tuned, along with most 5.25" diskettes.

**ONLY \$39.95** + \$3.00 shipping (US/Canada)

SofCap Inc.

P.O. Box 131

Cedar Knolls, NJ 07927

Visa (201) 386-5876 MasterCard

If you want the absolute best possible performance from your hard disk:

#### TUNE IT UP WITH H.D. TUNEUP

Disk Optimizer is TM SoftLogic Solutions. H.D. Tuneup is TM SofCap.

Circle no. 349 on reader service card.

#### Parallel Programming for "C"

#### **INTERWORK**

A Concurrent Programming Toolkit

Interwork is a "C" program library which allows you to write your programs as a set of cooperating concurrent tasks. Very useful for simulation, real-time applications, and experimentation with parallel programming.

#### **FEATURES**

- Supports a very large number of tasks (typically more than 100) limited only by available memory. Low overhead per task results in very fast context switching.
- Provides a full set of inter-task communication (ITC) facilities, including shared memory, locks, semaphores, blocking queues, and UNIX\*-style signals. Also has building blocks for constructing your own ITC facilities.
- Handles interrupts (DOS version) and integrates them into task scheduling. Supply your own interrupt handlers or block tasks on interrupts.
- Lets you trace task switches and inter-task communication.
- Comes with complete documentation including a user's manual and reference manual of commands.

Interwork is available for the following systems:

Hardware	Operating System	Price
IBM PC, XT, AT	PC-DOS 2.0 or later	\$129
IBM PC AT	XENIX*	\$159
DEC VAX* SUN III	UNIX 4.2BSD	\$249

PC-DOS version is compatible with DeSmet, Lattice, and Microsoft C compilers.

Please specify hardware and operating system when ordering. Shipping and handling included; COD orders add \$2.50. Send check or money order to:



#### **Block Island Technologies**

Innovative Computer Software

13563 NW Cornell Road, Suite 230, Portland, Oregon 97229-5892 (503) 241-8971

\*Trademarks: UNIX, AT&T Bell Laboratories, Inc.; XENIX, Microsoft, Inc.; VAX, Digital Equipment Corporation

Circle no. 263 on reader service card.

#### 32000 CROSS ASSEMBLER

#### Listing One (Listing continued)

```
bytbuf[ 2 ] = '0' + ( 1 % 10 );
                 bytbuf[ 3 ] = '\0';
                 adexpt[ adexct ] = &bytbuf[ 0 ];
                 adexln[ adexct++ ] = 1;
/* opopt 'j': uwb bits for movs, cmps, skps */
          if ( opopt == 'j' ) {
             p += 5;
             uwbbits( word, p );
                                    /* in case no paren */
             adexct = 0;
/* opopt 'k': config bits for setcfg */
          if ( opopt == 'k' ) {
             p += 5;
             cfgbits ( word, p );
/* opopt 'l': operand 1 becomes 4 bit value */
          if( opopt == '1' && j == 0 ) {
             adexct = 0;
                                    /* unjunk extensions */
             1 = -1;
             if( strcmp( word, "upsr" ) == 0 ) 1 = 0;
             if( strcmp( word, "upsr") == 0 ) 1 = 0

if( strcmp( word, "fp") == 0 ) 1 = 8;

if( strcmp( word, "sp") == 0 ) 1 = 9;

if( strcmp( word, "sb") == 0 ) 1 = 10;
              if( strcmp( word, "psr" ) == 0 ) 1 = 13;
if( strcmp( word, "intbase" ) == 0 ) 1 = 14;
              if( strcmp( word, "mod" ) == 0 ) 1 = 15;
              if( 1 == -1 ) error( 'p', word );
                 else {
                 p = &opbuf[ 5 ];
                  *p++ = '0' + (( 1 >> 3 ) & 1 );
                 *p++ = '0' + ((1 >> 2 ) & 1 );
                 *p++ = '0' + (( 1 >> 1 ) & 1 );
                 *p = '0' + (1 & 1);
                 p = &opbuf[ 0 ];
          1
/* odd length extension for movm, opopt 'o' */
           if( j == 2 && opopt == 'o' ) {
              if( adexct > 0 ) adexln[ --adexct ] = 1;
                 else error ('e', w);
              1 = value( adexpt[ adexct ] ) - 1;
              switch ( opsiz ) {
              case 'd' : 1 *= 4; break; case 'w' : 1 *= 2; break;
              bytbuf[ 0 ] = '0' + (( 1 / 100 ) % 10 );
bytbuf[ 1 ] = '0' + (( 1 / 10 ) % 10 );
              bytbuf[ 2 ] = '0' + ( 1 % 10 );
              bytbuf[ 3 ] = '\0';
              adexpt[ adexct++ ] = &bytbuf[ 0 ];
                   /* done operands */
       o = value( &opbuf[ 0 ] );
       l = strlen( opbuf );
/* Send as many opcode bytes as necessary */
       objout ( o % 256 );
       if(1 > 9) objout((o / 256) % 256);
       if(1 > 17) objout(o / 65536);
/* Send postbytes for scaled index mode */
       for( 1 = 0; 1 < opexct; ++1 )
          objout ( opexbt[ 1 ] );
/* Send addressing extensions.
```

```
adexln = length of extension word in bytes if +.
   If 0, it is a variable-length signed displacement.
   If -1, indicates code-relative */
/* If opcode ocnt was negative, last address extension is
code relative. */
      if(opcode[i].ocnt < 0)
         adexln[adexct - 1] = -1;
/* send extension words */
      for( j = 0; j < adexct; ++j ) {
         o = value(adexpt[j]);
/* if adexln[] negative, operand(s) code-relative.
   Note: on the 32000 you don't correct by adding 2 to
   codadr first */
         if(adexln[j] < 0) o -= ocodadr;
/* Compute variable-length signed displacement */
         if( adexln[ j ] <= 0 ) {
               ( o < 63 && o .
o = ( o & 0x7F );
/* one-byte */
            if( o < 63 && o > -64 ) {
            } else if( o < 8191 && o > -8192 ) {
               o = ( o & 0x3FFF ) + 0x8000;
               1 = 2:
            } else {
               o = (o & 0x3FFFFFFF )
                  + 0xC0000000;
               1 = 4;
         } else l = adexln[ j ];
/* address extensions are sent in lohi order */
         if(1 > 3 ) objout((o >> 24 ) & 0xFF);
         if(1 > 2 ) objout((o >> 16) & 0xFF);
         if(1 > 1 ) objout((o >> 8 ) & 0xFF );
         objout ( o % 256 );
      return 1:
   return 0:
/* Special to create extension word for register list */
/* Regbits may look like "r0" or may look like "[r0,r2]"
   or like "[r0-r7]". */
char *regbits ( src, dst, flg )
char *src, *dst, flg;
   int bits, reg, loreg, hireg;
  if( *src == '[' ) ++src;
  else error( '[', src );
                                /* strip parens */
      if( *src++ != 'r' ) error( 'r', src );
      reg = ( *src++ ) - '0';
     bits = bits | (1 << reg );
if( *src++ == '-' ) {
         loreg = reg;
         if( *src++ != 'r' ) error( 'r', src );
         hireg = ( *src++ ) - '0';
         if ( hireg < loreg ) {
            reg = hireg;
            hireg = loreg;
           loreg = reg;
                        /* swap if out of order */
         for( reg = loreg; reg <= hireg; ++reg )
           if( *src == ']' ) break;
                                   (continued on next page)
```

#### Modula-2

#### IBM PC/DOS Native Code Compiler

This is a full implementation of Niklaus Wirth's Modula-2 language. Our product is not an interpreter, but a true 8086 compiler, using state-of-the art techniques. A Unix-like "make" utility is included which provides automatic recompilation of modified source programs.

The code generator produces object module input for the DOS link utility. You may combine your Modula-2 programs with code from other languages such as Assembler. The software also operates on PC compatibles using MS/DOS. All the run time source code is included. None of the software is copy protected, and is fully supported and maintained by farbware. No royalties are charged for the use of the run time object. A complete and comprehensive reference manual is included in the purchase price. The manual is available separately for \$25.00.

Site licenses and quantity discounts are available.

#### \$89.95 Complete

farbware 1329 Gregory Wilmette, IL 60091 (312) 251-5310

Master Card and Visa Accepted

Circle no. 340 on reader service card.

# LEARN CON

#### with the video training course A Programmer's Introduction to C by Ray Swartz

Topics Covered: Data Types • Operators • Expressions • Loops • Conditionals • Input and Output Character Handling • Interactive Control · Arrays · Pointers · Switch

#### Reviewers' Comments:

"The tapes cover the major features of C and are aimed at professionals with a background in another computer language. The key to the package is that Swartz knows his subject and how to teach it." Dr. Lance Leventhal

"If you're looking for a way to learn C quickly and easily, this course is something you'll want to consider."

Doug Topham

Price: \$400 includes two video tapes (3½ hours) and 106 page manual.

**ASK ABOUT OUR FREE 15 DAY REVIEW** 

BERKELEY DECISIONS/SYSTEMS TO ORDER CALL (408) 458-0500

Circle no. 202 on reader service card

#### 32000 CROSS ASSEMBLER

#### Listing One (Listing continued)

```
/* if flg = 'f', save/enter, need to swap bit
  significance. The routine above constructed it in
  reversed order in the first place, because of the
  routine below */
  if( flg == 'f' ) {
      hireg = 0;
      for ( reg = 0; reg < 8; ++reg ) {
         hireg = ( hireg << 1 ) + ( bits & 1 );
         bits = ( bits >> 1 );
      bits = hireg;
/* now create a binary string for the extension.
  Note that bit significance becomes reversed again */
   for( reg = 0; reg < 8; ++reg ) {
 *dst++ = '0' + ( bits & 1 );
      bits = ( bits >> 1 );
   *dst++ = 'b';
                        /* add b for binary */
                      /* terminate */
   *dst++ = '\0';
   return dst;
/* put config bits into instruction */
cfabits ( src. dst )
char *src, *dst;
   char b[ 4 ];
   b[ 0 ] = '0';
   b[ 1 ] = '0';
   b[ 2 ] = '0';
   b[ 3 ] = '0';
   if( *src == '[' ) ++src;
                                 /* strip parens */
      else error('[', src);
   while( *src ) {
      switch ( *src++ ) {
      case 'c' : b[ 0 ] = '1';
           break;
      case 'm' : b[ 1 ] = '1';
            break:
      case 'f' : b[ 2 ] = '1';
      case 'i' : b[ 3 ] = '1';
      if( *src++ == ']' ) break;
   *dst++ = b[ 0 1:
   *dst++ = b[ 1 ];
   *dst++ = b[ 2 ];
   *dst = b[ 3 ];
/* put uwb bits into instruction */
uwbbits ( src, dst )
char *src, *dst;
   char b[ 3 ];
   b[ 0 ] = '0';
                         /* default = forward */
   b[ 1 ] = '0';
                         /* default = neither */
   b[ 2 ] = '0';
   while( *src ) {
      switch( *src++ ) {
```

```
case 'b' : b[ 2 ] = '1'; /* backward */
     case 'u' : b[ 0 ] = '1'; /* until match */
           b[ 1 ] = '1';
           break:
     case 'w' : b[ 0 ] = '0'; /* while match */
           b[ 1 ] = '1';
  *dst++ = b[ 0 ];
  *dst++ = b[ 1 ];
  *dst = b[2];
/* Check to see if the word begins an equate, and if it
  does, add the symbol to the symbol table. */
int isequate ( w )
char *w;
  char tempword[ 128 ];
  char *q, *cpystr();
  long int 1, getarg();
  q = cpystr( w, &tempword[ 0 ] );
                        /* get next word */
  if( strcmp( word, "equ" ) == 0 ||
    strcmp( word, "=" ) == 0 ) {
      1 = getarg();
                       /* get argument */
      addsymbol ( &tempword[ 0 ], 1 );
                        /* it was an equate */
      return 1;
  return 0;
                      /* we lost a word */
/* Get an argument value (for use above). */
long int getarg()
  long int value();
                        /* get next word */
   return value ( word );
/* copy string and return new ending address */
char *cpystr( src, dst )
char *src, *dst;
   while( *src ) *dst++ = *src++;
   return dst; /* return next address */
   return dst:
/* Calculate the value of a word. It may be a symbol, a
   constant, or a computed value (must be enclosed in
   parentheses.) */
long int value ( w )
   long int hexbin(), octbin(), bitbin(), decbin(), v;
   int lookup(), i, negate;
   char *q:
   char *wp[ 16 ];
   int wpcnt;
   negate = 0:
   if( *w == '-' ) {    /* Unary negation */
```

```
negate = 1:
      ++w;
    if( strcmp( w, "." ) == 0 )
   return codadr; /* . = code address */
if( strcmp( w, ".." ) == 0 )
      return asmadr; /* .. = assembly address */
   if( isdigit( *w )) {
   if( match( w, "*h" )) v = hexbin( w );
          else if ( match ( w, "*q" )) v = octbin ( w );
             else if ( match ( w, "*b" ))
                v = bitbin(w);
                else v = decbin(w);
   } else {
      if( *w == '(' ) {
                                 /* --- FORMULA --- */
          ++w;
                                  /* skip ( */
          q = w;
         while( *q ) ++q;
                                 /* find end of string */
          if( *q != ')' ) error(')', q);
             else *q = '\0';
                                 /* zap ) */
          iwparen = 0;
                                 /* no parens now */
          wpcnt = 0;
          while(1) {
                                 /* find beg of word */
             while( inword( *w )) ++w;
             if(! *w ) break;
            wp[ wpcnt++ ] = w; /* ptr to value */
             iwparen = 0;
                                 /* find end of word */
            while( *w && ! inword( *w )) ++w;
             if(! *w ) break;
             *w++ = '\0';
                                 /* terminate it */
            if( wpcnt == 16 ) {
  error( 'l', w ); /* too long */
         if(( wpcnt % 2 ) == 0 ) {
            error('v', w); /* must be odd */
            --wpcnt;
         v = value(wp[0]);
         for( i = 1; i < wpcnt; i += 2 ) {
  if( strcmp( wp[ i ], "+" ) == 0 ) {</pre>
               v += value( wp[ i + 1 ] );
               goto opdone:
            if( strcmp( wp[ i ], "-" ) == 0 ) {
               v -= value( wp[ i + 1 ] );
               goto opdone;
            if( strcmp( wp[ i ], "*" ) == 0 ) {
               v *= value( wp[ i + 1 ] );
               goto opdone;
            if( strcmp( wp[ i ], "/" ) == 0 ) {
               v /= value( wp[ i + 1 ] );
               goto opdone;
            error('o', wp[i]);
                                       /* unknown op */
opdone:
                                /* get around c/80 bug */
                                 /* --- PLAIN VALUE --- */
      } else {
         }
                                   (continued on next page)
```

#### C CODE FOR THE PC

source code, of course

-

GraphiC 3.0 hi-res color plots	\$300
Panache C Program Generator	\$125
QC88 C Compiler	. \$90
Concurrent C	. \$45
Coder's Prolog in C	. \$45
Biggerstaff's System Tools	. \$40
Translate Rules to C	. \$30
LEX	. \$25
YACC & PREP	. \$25
tiny-c interpreter & shell	. \$20
C Tools	. \$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409
(512) 258-0785

Circle no. 250 on reader service card.

Free shipping on prepaid orders

No credit cards

# Symbolic Debugger for Turbo Pascal

For PC-DOS and MS-DOS computers.

- Examine and modify variables during program execution.
- Set breakpoints at procedures, functions and labels by name.
- Variables are displayed according to their declared types.
- View Pascal source on the screen or send it to the printer while your program is running.
- · Line-by-line single step.
- Debug window allows debugging of programs without interfering with screen output.
- Much more, including machine-level debugging features.

Suggested retail price: \$59

Our mail order price: \$49 plus Shipping and Handling:

\$3 US/Canada, \$11 elsewhere.

VISA and MASTERCARD accepted.

Texas residents please add 6.125% sales tax.

Unsure? Send \$2 for a demonstration disc, refunded upon purchase.

Kydor Computer Systems
1701 Greenville Avenue Suite 505
Richardson, Texas 75081
(214) 669-1888

#### 32000 CROSS ASSEMBLER

#### Listing One (Listing continued)

```
if ( negate ) v = 0 - v;
  return v;
/* function for value() */
int inword(c)
char c;
  if( c == '(') ++iwparen;
                              /* special var for this */
  if( c == ')' ) --iwparen;
                               /* function */
  if ( iwparen ) return 0;
 if( c == ' ' ) return 1; /* is space */
  return 0:
/* --- SYMBOL TABLE LOGIC --- */
/* add new symbol to symbol table */
addsymbol (p, v)
char *p;
long int v;
   char *w, *cpystr(), *alloc();
   int i, lookup();
                     /* see if already known */
   i = lookup(p);
                      /* new symbol */
   if(i < 0) {
      i = symcnt;
                        /* count a new symbol */
      ++symcnt;
      symbol[ i ].snam = alloc( strlen( p ) + 1 );
      w = cpystr(p, symbol[i].snam);
   symbol[ i ].sval = v; /* update value in table */
/* lookup - returns symbol number or -1 if not found */
int lookup (p)
char *p;
   char *w;
   int i, j, k, found;
                        /* not found yet */
/* pass 1 - use linear search */
   if( pass == 1 ) {
      for ( i = 0; i < syment && ! found; ++i ) {
         w = symbol[ i ].snam;
         found = (strcmp(p, w) == 0);
   | else {
/* passes 2 and 3 - use binary search */
       j = ( symcnt + 1 ) / 4; /* step to use */
                                /* starting point */
       i = symcnt / 2;
      k = j + 1;
                                /* one-step count */
       while(1) {
         w = symbol[ i ].snam;
          found = strcmp(p, w);
         if ( found == 0 ) {
            found = 1;
            break;
         } else if ( found < 0 ) i -= 1;
            else i += 1;
         if(1 < 0) i = 0;
         if( i >= symcnt ) i = symcnt - 1;
```

```
j /= 2;
                                 /* halve step */
         if( j == 0 ) {
            if(k--=0) {
               found = 0:
                                /* not found */
               break;
            j = 1;
     }
  if(! found) {
     if( pass != 1 ) error( 'u', w );
     return -1;
  return i;
/* display error code */
error(c, p)
char c;
char *p;
  puts( "\n>>---> Error " );
  putchar( c );
puts( " at " );
  puts(p);
   ++errors:
/* sort symbols by shell sort */
sortsyms()
   int jump, done, k, 1;
   char *n;
   long int v;
                      /* set jmp to cnt of elements */
   jump = syment:
   while( jump > 0 ) {
      jump = jump / 2;
      while(1) {
         done = 1;
         for(k = 0; k < (syment - jump); ++k) {
            l = k + jump;
            if( strcmp( symbol[ k ].snam,
               symbol[ 1 ].snam ) > 0 ) {
               n = symbol[ k ].snam;
               v = symbol[ k ].sval;
               symbol[ k ].snam = symbol[ l ].snam;
               symbol[ k ].sval = symbol[ l ].sval;
               symbol[ l ].snam = n;
               symbol[ l ].sval = v;
               done = 0;
         if ( done ) break;
      }
   }
/* dump symbol table */
dumpsyms ()
   char *w;
   int i:
 long int v;
 puts ( "\nSymbol and Value\n" );
 for ( i = 0; i < syment; ++i ) {
    puts( symbol[ i ].snam );
    puts("=");
v = symbol[i].sval;
    puthex ( v >> 24, 0 );
    puthex(( v >> 16 ) & 0xFF, 0 );
                                    (continued on page 97)
```

#### Modula-2

When your project gets too big to manage in C, switch to Modula-2: the language designed for building large systems. And switch to REPERTOIRE, \*\* from PMI: low-level tools and high-level subsystems designed specifically for Modula-2

#### Partial list of utilities included in REPERTOIRE:

- DBMS: insertion, deletion and \* Screen Design/Display System: retrieval of keyed records; garbage collection and recovery of damaged files; variable-length records and fields; data and index stored together; convenient storage/retrieval of linked lists and unstructured text; nested fields; interactive filesearching expressions; full index kept in RAM for fast access.
- ★ Over 200 low-level routines.

for complete set of fully useable SYM and LNK files, but no source. Manual on disk

Upgrade to full source for . . . . \$79

Design full-color screens in any ASCII editor and display them instantly with an M2 procedure call. Screens obtain and check input, provide help, and scroll within windows. DMA, BIOS, or ANSI drivers selected at run time. Auto-mapping of mono/color attributes. Integrated windoworiented text editor. Not a code generator.

Thoroughly-indexed, 300-page manual

\$89 for full Modula-2 source code (590 K) and printed manual.

#### **OPTIONS:**

ModBase: an alternate DBMS fully compatible with Ashton-Tate's dBase III; create and access dBase III files from Modula-2, and vice-versa; disk-based B+ Tree Indexing system allows files with millions of

★ EXE2LNK: Allows Microsoft's MASM and any DOS linker to be used conveniently with Logitech's Modula-2/86.

ModBase: \$89 with full source

\$49 for all LNKs & SYMs.



The leading supplier of Modula-2 software components.

Compuserve: 74706, 262 4536 S.E. 50th • Portland, OR 97206

Call for free demo & documentation disk!

Circle no. 239 on reader service card.

#### MAC'S-A-MII

THE LOW-COST 2 MEGABYTE MEMORY UPGRADE BOARD FOR THE APPLE MACINTOSH.

Adds up to 1536K Contiguous RAM to 128K and 512K Macs. High Quality, Four-Layer, fully socketed daughter board.

All boards come assembled and tested with ONE YEAR WARRANTY. All RAM recognized by System, Finder and Switcher. RAMdisk included Reverts to 512K for programs which can't handle more memory.

Available in kit form or with guaranteed installation (requires soldering.) Run with eight programs in Switcher or with an 1800K+ RAMdisk! The ideal board for those who performed their own DDJ 512K upgrade!

		Kit	Installed
Board w/o RAM	only	\$195	\$255
128K to 512K	only	\$ 95	\$175
128K to 1 Meg	only	\$325	\$465
128K to 2 Meg	only	\$445	\$585
512K to 1 Meg	only	\$255	\$315
512K to 2 Meg	only	\$375	\$435
1 Meg to 2 Meg	only	\$120	\$120
Quiet piezoelectric fan	only	\$ 39	\$ 39
Patch for 64K ROMs	only	\$ 29	\$ 29

WA residents add 7.9% sales tax. Include \$5 per board for Shipping and Handling, \$20 if you send your whole Mac. Allow 2 weeks for personal checks to clear. Add \$1.90 for C.O.D.

Board w/o RAM

ILL

1314 N.E. 43rd Suite 216 Seattle, WA 98105

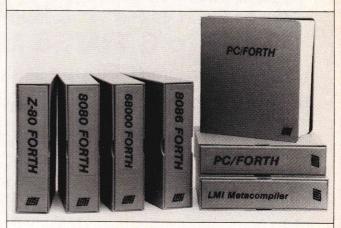
Call (206) 547-4779 for more information.

Dealer and Installer inquiries welcome.

Prices subject to change without notice.

Mac's-a-Million is a trademark of Sophisticated Circuits, Inc. Macintosh is a trademark licensed to Apple Computer, It Circle no. 354 on reader service card

#### TOTALCONTROL with LMI FORTH



#### For Programming Professionals: an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

#### For Development: Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

#### For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

#### For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- · Can generate ROMable code

#### Support Services for registered users:

- Technical Assistance Hotline
- · Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.



#### Overseas Distributors.

Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665 UK: System Science Ltd., London, 01-248 0962 France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16 Japan: Southern Pacific Ltd., Yokohama, 045-314-9514 Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

#### **TURBO**

#### SCREEN/APPLICATION GENERATOR **BE 3 to 6 TIMES MORE PRODUCTIVE**

For \$99.95 receive 6 Floppy Disks & Manual. DDJ Special: \$69.95

#### **TURBO ISAM MASTER**

\* \* \* \* \* \*

TURBO MASTER helps you develop your functional specs and then allows you to QUICKLY prototype a validation model of your system. TURBO MASTER can then generate a super-fast Turbo Pascal Program that features advanced screen input and control, a professional control menu, and database functions.

**MASTER DATABASE PROGRAM - Generates** Pascal Program Code for the following functions

-Add a Record - Allows both duplicate & unique only keys. Change the up to 48 fields comprising the primary key and the up to 7 secondary keys at any time during input, and the validity of the keys are checked and the ISAM key files are automatically adjusted.

-Delete a Record - Shows the record to be deleted on the screen and allows you to change your mind about deleting, and then adjusts the keys files automatically for the up to 8 keys. The disk space is reused

automatically by the programs.

-Edit a Record - with a key change allowed. -Search Database by Key - The ability to display the keys on the screen or printer in sorted order.

DATA BASE RECOVERY PROGRAM. This program recovers your database if it corrupted by a power outage or certain hardware failures.

MULTIPLE ISAM FILES - Can be used at the same time in a single program. The generator also produces context sensitive instructions on how to integrate the generated program into your main program.

**DOCUMENTATION** - Print screens and ISAM specifications. Also inline program documentation is generated.

#### **TURBO MENU MASTER**

GENERATES 'Ready to Run' Turbo Programs with Dedicated Screens or Windows.

FEATURES: Has Menu Data Base so it's easy to modify menus.

#### PROVIDES SELECTION BY:

- -Press of a number
- -Press of a function key
- -Press the high lighted letter

-Use the arrow keys

ADAPTIVE SCREEN COLORS - Different screen colors are automatically used for color or monochrome display. This allows you to provide a better interface for the user

#### INTERACTIVE MENU BUILDER

- -Allows for the automatic entry and reorder of
- Offers easy color selection
- -Allows for choice of procedure, chain, execute or comment code generation for each
- -Has startup Menu that YOU design.
- -Provides for the integrated Display and control of the Key Lock Status.

#### TURBO RESIDENT SCREEN CAPTURE

which allows you to capture Text Screens from any running program.

#### **TURBO SCREEN MASTER**

GENERATES 'Ready to Run' Mulit-screen programs with advanced field definition.

USER SPECIFIED PROCEDURES can be called before and after data entry for each field. This allows SCREEN GENERATION THAT IS INDEPENDENT OF USER INSERTED PRO-GRAM STATEMENTS.

YOU CAN CUSTOM PROGRAM special field edits and processes which Screen Master automatically includes into the screen program. This allows for "On the Fly FUNCTION KEY AND WINDOW ROUTINES.

**RECORD FORMATS** and initialization routines for Structures and Arrays are automatically

generated.

• CAN SPECIFY a Protected Field to be automatically redisplayed if its value changes.

TYPES INCLUDE Strings, Yes/No, Time, Male/Female and Numbers

FIELDS CAN BE STORED (declared) as Boolean, Integer, Real, Character, or String.

COMPARISON WITH OTHER PRODUCTS	TURBO SCREEN MASTER	SCREEN SCULPTURE Ver - 1.01
Full support for structures, arrays and Declarations specifications Full support for user written pro-	YES	NO
cedures, function keys & help screens	YES	NO
Etch-A-Sketch Border Drawings	YES	NO
Point and Paint color interaction	YES	YES
Border color control	YES	NO
User defined valid character sets	YES	NO
Display of Caps/Num Lock Status Optional realtime initialization of	YES	NO
data/timo	VES	NO

#### TURBO RESIDENT ISAM

Replaces Borland's Toolbox ISAM method

· Eliminates 8k of code space that Borland's Turbo Toolbox ISAM requires in your program.

Puts the Key Buffer Data Area outside of your program's data space.

Speeds Compling

· Eliminates errors caused by inconsistant ISAM specifications between ISAM files.

Increases program loading time.

Reduces object program size.

Offers SUPER-FAST record creation and lookup. For example, it can CREATE over 20 single key records PER SECOND on an IBM AT running at a 9 meg clock rate.

#### **OUR USERS REPORT**

"Since Fall of 85, I have generated over 300 program modules with it and find it to be just what I needed. Most of all the modules represent 5000 to 8000 lines of Pascal Code" Oner Systems

"By being able to produce a 21 screen and menu control demo so quickly helped me obtain the contract.'

"Speeded up my screen development by 6 times" Elexor Associates.

"Has many of the features of the Super Mini development tools costing \$10,000." Applied Micro Systems.

"We developed 3 Vertical Market Applications in the 6 months we had your system." Absolute

#### \* BTRIEVE INTERFACE MODULE

\* Allows full multiuser record locking and \* ¥ Automatic file recovery for the industry's ★

¥ most popular LANs. Works with the in- ★ ¥ dustry's leader of professional databases ★

\* for multiuser LAN's (optional).

Requires Btrieve by SoftCraft, Inc. \*\*\*\*\*\*\*\*\*\*

#### **TURBO MASTER TOOLS**

The Turbo Toolkit Master includes procedures for full control over all screen attributes, advanced string functions, automatic control of multiple help screens, saving and retrieving screens to RAM buffers. Caps/Num/Scroll/Lock control procedures, and report procedures.

#### RISK FREE TRIAL

Try the demo package included for 30 days. If not pleased return for a full refund.

Credit Card Orders Call:

#### 1-800-821-9503

In Florida (800) 342-0137 For Other Information Call (305) 892-5686

NO ROYALTIES on Generated Programs

YES, ENCLOSED IS \$69.95

#### For

**Turbo Master Tools** Turbo Menu Master **Turbo Screen Master** Turbo ISAM Master **Turbo Screen Capture** Resident ISAM

Systems Requires: IBM PC,XT,AT or 100% Compatible 246K MS DOS 2.0 or Higher, Turbo Pascal 3.0 - 2 DD/DS

US orders add \$7.50 S&H

Name:

All foreign orders add \$15 per product ordered.

City:	Service Service		
State:			
Zip			

C.O.D.'s will be accepted. Outside USA: make payment by

bank draft, payable in US dollars drawn on a US bank.

Turbo Pascal & Turbo Database Toolbox are trademarks of Borland International, IBM is a trademark of International Business Machines. MS-DOS is a trademark of Microsoft. Btrieve is a trademark of Softcraft, Inc. Screen Sculpture is a trademark of the Software Bottling Co. of New York. © 1985 Hawaiian Village Com-

HAWAIIAN VILLAGE COMPUTER SOFTWARE 1109 Pennsylvania Ave., St. Cloud, Florida 32769

#### 32000 CROSS ASSEMBLER

#### Listing One (Listing continued)

```
puthex(( v >> 8 ) & 0xFF, 0 );
      puthex( v & OxFF, 0 ); /* print value */
     putchar('\n');
/* Match string. If match, returns 1; else returns 0.
   Ambiguous values from the matches are saved and
   pointed to by the array of char pointers ambig[], so
  they can be checked later. */
int match ( w1, w2 )
char *w1, *w2;
   char c:
   char *next ambig;
   next ambig = &ambig buffer[ 0 ]; /* init ambig buff */
                                   /* ambigs so far */
   while( *w1 ) {
     c = *w2++;
if( c == '*' ) {
         ambig[ ambcnt++ ] = next ambig;
        while( *w1 && *w1 != *w2 )
        *next_ambig++ = *w1++;
if(! *w1 && *w2) return 0;
        *next ambig++ = '\0'; /* terminate this ambig */
      } else if ( c == '?' ) {
           ambig[ ambcnt++ ] = next_ambig;
           } else if( c != *w1++ ) return 0;
   return 1:
int gword()
   char *p, *q;
   char c, gchar();
   p = &word buffer[ 0 ];
   c = ' ';
   while( isdelim( c )) c = gchar();
   while(! isdelim(c)) {
     *p++ = tolower( c );
      c = gchar();
   *p = '\0'; /* terminate word */
   word = &word buffer[ 0 ];
   return 1:
/* is the character a delimeter? */
isdelim(c)
   if( paren || quote || brack )
  return 0; /* not a delim */
   return 1;
   return 0:
/* get next char from source file */
char gchar ()
   char c, getch();
                        /* get char from file */
   c = getch();
```

```
if( c == '\'' ) quote = ! quote;
if( c == '"' ) quote = ! quote;
   if( c == '(') ++paren;
if( c == ')') --paren;
   if( c == '[' ) ++brack;
   if( c == ']' ) --brack;
   if( ! quote && ! paren && ! brack ) {
      while( c == ';' ) {
                                    /* ;comment\n */
         while ( getch () != '\n' );
          c = getch();
   return c;
puts (p)
char *p;
   while( *p ) putchar( *p++ );
/* --- source file routines --- */
char getch()
   while( inpcnt == 0 ) {    /* if input buf
   listpr();    /* print listing line */
                                  /* if input buf empty, */
      inpload():
                          /* reload input buffer */
   --inpent:
   return(inpbuf[inpptr++]);
inpload()
   char c, getc();
   inpcnt = 0;
   inpptr = 0;
   while((( c = getc( fasm )) != '\n' ) && ( c != EOF )) {
      inpbuf[inpcnt++] = c;
       if( listcp < 81 ) listline[ listcp++ ] = c;</pre>
   inpbuf[inpcnt++] = '\n';
/* --- listing file routines --- */
listnl()
                  /* list new line */
   int i:
   if ( pass != 3 ) return;
   for( i = 0; i < 26; ++i ) listline[ i ] = ' ';
   for( i = 26; i < 81; ++i ) listline[ i ] = '\0';
   listop = 0; /* flag to cause addr output */
   listcp = 26;
lbvt(b)
                  /* put object byte in list file */
unsigned int b;
   char c:
   if ( pass != 3 ) return;
   c = ((b / 16) % 16) + '0';
   if(c > '9') c += ('a' - ':');
listline[listop++] = c;
   c = ( b % 16 ) + '0';
if( c > '9' ) c += ( 'a' - ':' );
listline[ listop++ ] = c;
```

#### 32000 CROSS ASSEMBLER

#### Listing One (Listing continued)

```
if( listop > 24 ) listpr(); /* print list line */
listpr()
                /* print list line */
   if ( pass != 3 ) return;
   putchar('\n');
   puts ( listline );
   listnl():
1
/* --- object file routines --- */
objout ( c )
char c;
               /* incr asmadr, codadr. DON'T incr*/
/* objadr, it is addr of 1st byte */
   asmadr++;
   codadr++;
   if (pass!=3) return; /* skip if not last pass */
objbuf[objcnt++] = c; /* put new byte in buffer */
if (objcnt == 32) objflush();
   if( listop == 0 ) {
                                    /* print address? */
       lbyt ( asmadr / 16777216 );
       lbyt (( asmadr / 65536 ) % 256 );
       lbyt (( asmadr / 256 ) % 256 );
       lbyt ( asmadr % 256 );
      listop = 9;
                         /* send byte to listing too */
   lbyt (c);
}
objflush()
   int i. cksum:
   if ( pass != 3 ) return; /* just in case we get here */
   cksum = 0:
   if( objent > 0 ) {
      putc( ':', fobj );
       puthex ( objent, fobj );
      puthex( objadr / 256, fobj );
puthex( objadr % 256, fobj );
       puthex ( 0, fobj );
       cksum =
         objent + ( objadr / 256 ) + ( objadr % 256 );
       for( i = 0; i < objent; ++i ) {
         puthex( objbuf[ i ], fobj );
          cksum += objbuf[ i ];
      puthex (0 - cksum, fobj);
      putc('\n', fobj);
   objadr = asmadr;
   objent = 0;}
puthex (b, c)
int b, c;
   int v;
   v = ( b & 0x00F0 ) >> 4;
if( v > 9 ) v += 'A' - 10; else v += '0';
   putc( v, c);
   v = (b & 0x000F);
    if( v > 9 ) v += 'A' - 10; else v += '0';
    putc( v, c );
 long int hexbin(p)
 char *p;
    long int v;
    v = 0:
```

```
while( *p ) {
      if( isdigit( *p )) v = ( 16 * v ) + *p++ - '0';
else if( *p >= 'a' && *p <= 'f' )
            v = (16 * v) + *p++ - 'a' + 10;
            else ++p;
  return v;
long int octbin ( p )
char *p;
  long int v;
  v = 0:
  while( *p ) {
     if( *p >= '0' && *p <= '7' )
        v = (8 * v) + *p++ - '0';
         else ++p;
  return v;
long int bitbin ( p )
char *p;
   long int v;
  v = 0:
   while( *p ) {
      if( *p == '0' || *p == '1' )
         v = (2 * v) + *p++ - '0';
         else ++p;
   return v:
}
long int decbin (p)
   long int v;
  v = 0;
  while( *p ) {
     if( isdigit( *p )) v = ( 10 * v ) + *p++ - '0';
        else ++p;
  return v;
#include "stdlib.c"
```

**End Listing** 



#### From B C Associates — SimpleNET™

#### SIMPLY THE LOWEST COST LAN

Yes, it's true, now you can take control of your data handling problems and implement your own PC local area network for only \$99.00 per station (plus software, power supply and cables).

Are you tired of carrying around a box of diskettes just to transfer information among your many PC systems in your office? You've probably looked into networks, but the high cost of such systems kept you away.

Now there's SimpleNET™. A truly low cost/medium performance alternate to the high priced systems. SimpleNET uses a small interface module which attaches to your PC systems and a PC Network (DOS 3.x) compatible network BIOS program. The interface allows up to 32 users to be connected via a single interface cable with a maximum cable length of 1.2 kilometers (how about 4000 feet?). The software interface is compatible with DOS 3.x and the new PC Local Area Network Program available from your IBM dealer.

#### <u>SimpleNET Basic System\* — For up to 4 users</u>

- Interface/Power supply module. (One power supply module is capable of driving 8 stations.)
  - User Interface modules
  - Cable Package

- Network BIOS Software
- Installation/Operations Manual

Only \$69500 Complete

#### PROGRAMMERS AND SOFTWARE DEVELOPERS - LOOK AT THESE PRODUCTS! **NO ROYALTIES REQUIRED**

#### **ASMLIB** The Programmer's Library

- A Multipurpose set of over 200 Assembly Language sub routines supplied in the form of a linkable library.

- Floating point math and trig routines with 8087 support.
- Graphics on EGA, herc. and CGA. 00 Floating point math and triggraphics. Installable keyboard activated programs are easily written with ASMLIB's special functions.
- Plus much more.Supplied with complete source code.

#### Only \$14900 Complete

#### asmTREE The Programmer's B+Tree Data File **Management System**

- A complete single/ multiuser database management system written entirely in Assembly Language grows the Lattice "C" or Assembly Language programmer dese capabilities.
- Up to 256 users.
- Up to 256 index and data ties.

  Multiple key types

- Multiple key types. SC Multiple indices per index file. Duplicate and variable length keys. Virtual file handling Plus vach, much more

- Supplied with complete source code.

#### Only \$39500 Complete

GenericGL - Generic general ledger package can be used by any program...UDS...\$295.00

FSEdit - Full sreen edit package by UDS...\$49.95

#### **REALIA COBOL USERS!**

FPLIB - Floating point library package with 8087 support and trig functions...\$149.00

Full Money Back Guarantee

#### **BC ASSOCIATES**

3261 No. Harbor Blvd., Suite B Fullerton, CA 92635

1-800-262-8010

in Calif. Call (714) 526-5151

inclosed please find my UCheck	□ Money (	order	for	\$
Please send the following:				

OTY SimpleNET Basic 4 user...by UTE ...

asmTREE database development system ...... \$395.00 each

FPLIB Realia COBOL Floating point pkg ...... \$149.00 each All prices include UPS shipping within continental United

States. Outside U.S. please add \$10 per package. Calif residents please add 6.5% sales tax. Total



FSEdit full screen editor ...



#### C CHEST

#### Listing One (Text begins on page 104.)

```
#include <stdio.h>
  #include <fcntl.h>
  #include <getargs.h>
5
           EXEPRINT C
                             Either print or modify the exe file header:
           exe file
                             Print the contents of a file's EXE header
 8
           exe -mN file
                             Modify the exe header so that N bytes of memory
                             are allocated for the combined bss/stack/heap area. If N is smaller that the required minimum
10
11
                              (bss + stack size) then it's rounded up. The
                             largest permitted value of N is 65,535. Use -ml for the minimum possible heap.
12
13
                             Modify the exe header so that N bytes of stack are used. If necessary, increase the bss/stack/heap size to accommodate the new stack. (The
            exe -sN file
14
15
                             bss/stack/heap won't be made smaller, however).
17
18
20 typedef unsigned short word; /* 2-byte unsigned number
22 typedef struct
23
            word
                     signature;
                                      /* Length of load module image % 512
25
            word
                     image len;
                                      /* File size in 512-byte units
                     file size:
26
            word
27
                                      /* Number of relocation table items
            word
                     num reloc:
                                      /* Size of the neader in paragraphs
/* min size of data area above program
/* max size of data area above program
28
            word
                     header size;
29
                     bss min;
            word
30
            word
                     bss max;
                                      /* displacement in para. of stack seg.
                     stack disp;
            word
                                      /* Initial SP register contents
32
                     init sp;
            word
                                      checksum;
            word
34
            word
                     init_ip;
                     code_disp;
first reloc;
35
            word
36
            word
                                       /* overlay number.
                     overlay;
39 EXE HEADER;
                     Hsize = 0 , Ssize = 0 ;
42
43 ARG
            Argtab[] =
44 (
            { 'm' , INTEGER, &Hsize, "Set miminum heap size to <num>" }, { 's' , INTEGER, &Ssize, "Set stack size to <num>" }
45
46
47 };
49 #define TSIZE (sizeof (Argtab) / sizeof (ARG))
52
53 usage ()
54 (
            fprintf( stderr, "exe [-ms[<num>]] file\n" );
55
56
57 }
            exit (1):
 59
 60
 61 main(argc, argv)
62 char **argv;
 63 {
 64
            EXE HEADER
 65
                             fd;
 66
            unsigned
                             numpara, ostack, odata;
 67
            argc = getargs ( argc, argv, Argtab, TSIZE, usage );
 68
 69
 70
71
72
            if( argc != 2 )
                     fatal_err("exe: exactly one file name required\n");
 73
74
            76
            78
79
 80
 81
            if ( Hsize )
                  83
 84
 85
                      2) h.bss.max, the maximum heap size, gets either the
 86
                                   current minimum or the specified size,
 87
                                   whichever is larger.
                     3) write out the modified header.
 88
 89
 90
91
92
                  numpara = Hsize/16 + (Hsize % 16 != 0) ;
 93
                  h.bss_max = (numpara<h.bss min) ? h.bss min : numpara; /* 2 */
 94
                 lseek ( fd, OL, O );
                                                                               /* 3 */
```

(continued on page 102)

# Now You Know Why BRIEF is BEST

"BRIEF is simple to learn and use and extremely sophisticated."

PC Magazine, July 1986

# The Program Editor with the <u>BEST</u> Features

Since its introduction. BRIEF has been sweeping programmers off their feet. Why? Because BRIEF offers the features MOST ASKED **FOR** by professional programmers. In fact, BRIEF has just about every feature you've ever seen or imagined. including the ability to configure windows, keyboard assignments, and commands to YOUR preference. One reviewer (David Irwin, DATA BASED ADVISOR) put it most aptly, '(BRIEF)...is quite simply the best code editor I have seen."

#### **COMPILER SUPPORT**

No matter what compiler you have, it will run inside BRIEF. If errors occur during compilation, the supplied macros place your cursor on the line with the first problem and display the compiler's message. After you make your corrections, you skip to the next error with one keystroke. BRIEF automatically moves your cursor to the right place, even if you've added or deleted lines.

BRIEF is preconfigured (using the built-in macro language) for the Microsoft Macro Assembler v 4.0, and the Microsoft, Computer Innovations, Lattice, and Wizard C compilers. If you use another product, you can modify the macros to support it.

#### Every Feature You Can Imagine

Compare these features with your editor (or any other for that matter).

- FAST
- Full UNDO (N Times)
- Edit Multiple Large Files
- Compiler-specific support, like auto indent, syntax check, compile within BRIEF, and template editing
- Exit to DOS inside BRIEF
- Uses all Available Memory
- Tutorial
- Repeat Keystroke Sequences
- 15 Minute Learning Time
- Windows (Tiled and Pop-up)

- Unlimited File Size –(even 2 Meg!)
- Reconfigurable Keyboard
- Context Sensitive Help
- Search for "regular expressions"
- Mnemonic Key Assignments
- Horizontal Scrolling
- Comprehensive Error Recovery
- A Complete Compiled Programmable and Readable Macro Language
- EGA and Large Display Support
- Adjustable line length up to 512

#### Program Editing YOUR Way

A typical program editor requires you to adjust your style of programming to its particular requirements - NOT SO WITH BRIEF. You can easily customize BRIEF to your way of doing things, making it a natural extension of your mind. For example, you can create ANY command and assign it to ANY key even basic function keys such as cursor-control keys or the return key.

#### The Experts Agree

Reviewers at BYTE, INFOWORLD, DATA BASED ADVISOR, and DR. DOBB'S JOURNAL all came to the same conclusion – BRIEF IS BEST!

Further, of 20 top industry experts who were given BRIEF to test, 15 were so impressed they scrapped their existing editors!

NOT COPY PROTECTED

Solution Systems \*\*

MONEY-BACK GUARANTEE

Try BRIEF (\$195) for 30 days – If not satisfied get a full refund.

TO ORDER CALL (800-821-2492)

SOLUTION SYSTEMS, 335-D WASHINGTON ST., NORWELL, MA 02061, 617-659-1571

BRIEF is a trademark of UnderWare



A comprehensive, indexed **Directory of Public Domain** C Source Code is now available.

To order, send \$10 (U.S.) to:

#### The C Users' Group

Post Office Box 97 McPherson, KS 67460 (316) 241-1065

Write or call for more information on CUG membership.

Circle no. 181 on reader service card.

#### PROMPT DELIVERY!!! SAME DAY SHIPPING (USUALLY) QUANTITY ONE PRICES SHOWN for NOV. 16, 1986

	OUTSIDE (	OKLAHON	A: NO SA	LES TAX	
Zenith 150, s; hp Vectra		YNAM 000Kx1 64Kx4	100 ns 150 ns	\$40.00 3.75	\$10.00
: Zeni lus; hp	41256 41256	256Kx1 256Kx1	100 ns 120 ns	4.85 2.95	8Mhz 8Mhz
rte MOTHERBOARD KITS: Zeni XT, Compaq Portable & Plus; hp	41256 41128 4164	256Kx1 128Kx1 64Kx1	150 ns 150 ns	2.55 4.99 1.35	V20 V30
RBOAF	27512	EPR 64Kx8	250 ns	\$22.00	\$125.00
МОТНЕ , Сомра	27C256 27256 27128	32Kx8 32Kx8 16Kx8	250 ns 250 ns 250 ns	5.85 5.25 3.95	5 Mhz SI 8Mhz SC
δÖ	27C64 2764	8Kx8 8Kx8 STATIO	200 ns 250 ns	4.75 3.75	5 2-7
640 IBM	43256L-12 6264LP-15		120 ns 150 ns	\$21.75 2.99	8087 8028

OPEN 61/2 DAYS, 7 AM-10 PM: SHIP VIA FED-EX ON SAT. SUNDAYS & HOLIDAYS: SHIPMENT OR DELIVERY, VIA U.S. EXPRESS MAIL

SAT DELIVERY

SAT DELIVERY

MasterCard/VISA or UP 10 Memory

Factory New, Prime Parts JIF 10 Memory

MICROPROCESSORS UNLIMITED, INC.

24,000 S. Peoria Ave.

(918) 267-4961

No minimum order

No minimum order

A suparace edia.

Circle no. 105 on reader service card.

#### Enhance Your Turbo Pascal™ Programming

#### TURBO-JET

- Ultra Fast Screen Read and Display
- Advanced String and Numeric Formatting Advanced File and Keyboard Handling
- Subdirectory Utilities
- Over 100 Files Included!

Pascal Source Code included with all Routines Routines Crafted in Assembly Language No Royalties for Program Use

Give Programs a Professional Look Don't Pay More For Less! Dealer Inquiries Invited

#### TURBO-JET, Only \$39.95

Add \$3.00 for Postage & Handling NY Residents add sales tax TOC Business Solutions, Inc. P.O. Box 129 Old Westbury, N.Y. 11568 MC/VISA (516) 795-2800

Circle no. 345 on reader service card.

```
Listing One (Listing continued, text begins on page 104.)
                   write( fd, (char *) &h, sizeof(h) );
   98
   99
             if ( Ssize )
  100
                           ostack = number of paragraphs in original stack odata = number of paragraphs of data. numpara = number of paragraphs in new stack.
  101
                   /* 1)
  102
  103
                           modify stack size.

Adjust the size of the stack+data area as appropriate.

write the modified header out to the file.
  104
                    * 4)
  105
                    * 5)
  106
                    * 6)
  107
  108
                   ostack = h.init_sp/16 + (h.init_sp % 16 != 0) ;
odata = h.bss_max - ostack ;
numpara = Ssize716 + (Ssize % 16 != 0) ;
  109
                                                                              /* 1 */
  110
  111
  112
  113
                   h.init sp = Ssize :
                                                                              /* 4 */
  114
  115
116
                   h.bss min = odata + numpara;
                                                                              /* 5 */
  117
                   if ( h.bss min > h.bss max )
                           h.bss max = h.bss min;
  119
                   lseek( fd, OL, O );
write( fd, (char *) &h, sizeof(h) );
                                                                              /* 6 */
  120
  121
  122
  123
  124
             print_hdr(&h);
close(fd);
  125
  126 }
  127
  128 /
  130 print_hdr(h)
131 EXE_HEADER
  132 {
             134
  135
             printf("%6d (0x%04x): ", h->image_len, h->image_len);
printf("Length of image mod 512\n");
  136
  137
  138
             139
  140
  141
             printf("%6d (0x*04x): ", h->num reloc, h->num reloc);
printf("Number of relocation table entries\n");
  142
  143
  144
             145
  146
  147
  148
             printf("%6u (0x%04x): ", h->bss min, h->bss min );
printf("Min. memory above program (paragraphs) = %lu bytes\n"
  149
  150
  151
                                               (unsigned long) h->bss_min * 16);
  152
             153
  154
  155
  156
             printf("%6d (0x%04x): ", h->stack disp, h->stack_disp);
  157
             printf("Displacement (paragraphs) of stack within load module\n");
  159
             160
  161
             163
  164
             printf("%6d (0x%04x): ", h->init ip, h->init ip);
printf("Initial value of the PC (IP) register\n");
  167
             printf("%6d (0x%04x): ", h->code_disp, h->code_disp);
  169
             printf("Displacement (paragraphs) of code seg. within load module\n");
  170
   171
             172
  173
174
  175
             printf("%6d (0x%04x): ", h->overlay, h->overlay);
printf("Overlay number.\n");
  176
  177 }
                                                                      End Listing One
  Listing Two
    1 #include <stdio.h>
2 #include <stdarg.h>
    4 fatal_err( fmt )
5 char *fmt;
                                      /* Print an error message to stderr and */
                                      /* then exit.
     6 1
               va_list args;
va_start(args, fmt);
               vfprintf( stderr, fmt, args );
   10
               exit(1);
                                                                      End Listing Two
   11 1
```

#### **Listing Three**

```
#include <stdio h>
                                                     NRTRAV.C: A non-recursive binary */
                                                     tree traversal routine that uses */
     typedef struct n
                                                     the link-inversion method.
               int
                             tag;
*left;
              struct
               struct n
                             *right;
  8
              char
 10 NODE:
 11
 12
    #define print (nodep)
                                  printf( "%s ", (nodep)->key );
 15
 16
 18 descend_left ( pres, prev )
19 NODE **pres, **prev;
                                                      /* Descend left till we can't
                                                      /* go any farther, reversing
 21
              register NODE
                                 *next:
              while ( next = (*pres)->left )
 24 25
                         (*pres) -> left = *prev;
 26
                         *prev
 27
                                         = next;
                        *pres
 28
 29 1
 30
 31
 32
 33 descend right ( pres, prev )
34 NODE **pres, **prev;
                                                         Descend right one node,
                                                      /* reversing links.
 35
                                                     /* Return 0 if we couldn't go.
 36
              register NODE
 37
 38
              if(!(next = (*pres)->right))
    return 0;
 39
 40
 41
               (*pres) -> tag = 1;
(*pres) -> right = *prev;
*prev = (*pres);
 42
 43
 44
               *pres
                                = next;
 45
              return 1:
 46 }
 47
 48
 49
 50 trav ( pres )
 51
    NODE
            *pres:
 52
 53
              NODE
                        *prev = NULL, *next;
 54
 55
              do
56
57
                        descend_left(&pres, &prev);
              print( pres );
} while( descend right( &pres, &prev) );
58
 59
 60
61
              while ( prev )
62
 63
                        if ( prev->tag == 0 )
                                                                        /* go up from a */
/* left child */
 64
65
                                               = prev->left;
66
                                 prev->left =
                                                 pres;
                                 pres
                                                 prev;
68
                                               = next;
                                 prev
69
70
                                 while(1)
                                                                        /* and back down */
71
72
                                           print( pres );
if( !descend_right(&pres, &prev) )
73
74
                                                    break;
75
76
77
                                           descend_left ( &pres, &prev );
                                 }
79
                       else
                                                                        /* go up from a
80
                                                                        /* right child
81
                                 next
                                                = prev->right;
82
                                                = 0;
                                 prev->tag
83
                                                = pres;
                                 pres
                                               = prev;
85
                                 prev
                                                 next:
86
```

**End Listings** 

#### **#1 C interpreter**



# The professional C development environment

Your C compiler creates great final code . . . but as a programming tool, it's too, too slow. With C-terp you can edit, debug, and run without the wait. Nothing, but nothing, is faster for developing professional C programs.

#### Choose the perfect C-terp companion for your C compiler

C-terp/Microsoft C-terp/XENIX
C-terp/Lattice C-terp/Aztec
C-terp/Mark Williams C-terp/C86

Link in all your compiler's functions, your own functions, add-on libraries, assembly routines, and data objects. Get instant access to everything in the C-terp interactive environment.

#### Only C-terp offers all this and more

- Full K&R with common ANSI enhancements
- Source level interactive debugging
- Software paging for your big jobs
- Complete multi-module support
- Run-time pointer checking
- Unsurpassed reconfigurable screen editor
- Dual display and full graphics support
- Large model Call-in

#### ORDER C-terp TODAY (specify compiler)

C-terp runs on IBM PC, AT or compatibles.

#### Price:

MS-DOS 2.x and up - \$298, Xenix System V 286 - \$498 MC, VISA, COD 30-day money-back GUARANTEE

Trademarks: C-terp (Gimpel Software), C86 (Computer innovations), Lattice(Lattice, Inc.), Xenix, Microsoft, MS-DOS (Microsoft, Inc.), Aztec (Manx Software), Mark Williams (Mark Williams Company), IBM (international Business Machines, Inc.)

#### GIMPEL SOFTWARE

3207 Hogarth Lane, Collegeville, PA 19426

(215) 584-4261

#### **Shrinking .EXE File Images**

It's been pointed out to me that the shell occupies much more memory at run time than is actually needed. Fortunately, the problem is easy to fix without having to recompile, and the techniques used are applicable to all .EXE files.

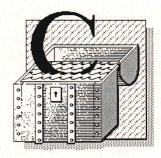
The problem has to do with how .EXE files are loaded into memory by MS-DOS and with how memory is used by *malloc()* and *free()*. (See this month's Flotsam and Jetsam, page 108, for a description of memory organization within a C program.)

The MS-DOS loader reads the text and data segments from the disk and then allocates all remaining memory for the bss segment, stack, and heap, even if the program is a small-model program that couldn't possibly use all that memory. When an .EXE file is loaded, DOS can reduce this default to an amount of memory specified in a header found at the beginning of the file (in the first 14 words). This amount has to be large enough to accommodate the entire bss and stack segments. The heap, however, doesn't need to be allocated at load time because more memory is requested from DOS if the heap isn't large enough when malloc() is called, thereby increasing the size of the run-time image.

The .EXE file header is initialized by the linker so that all available memory is allocated to the current program. Most small-model programs will then reduce this amount to a 64K

#### by Allen Holub

combined data/bss/stack/heap area as part of the boot process. Unfortunately, 64K is always allocated, whether or not you need it. The shell, in its released form, has this 64K data space allocated to it, even though it only needs about 3K for static data and another 3K for the heap. Conse-



quently it takes up almost 90K of memory rather than the 50K or so that's actually needed.

The automatic assignment of 64K can be circumvented by having the linker put a more reasonable number into the .EXE file header (by using the /CP or /STACK command-line options). It's not always convenient to relink, however, especially if you don't have the original source or object modules. Fortunately, the size of the run-time image can be reduced by doing nothing more than changing a couple of numbers in the .EXE file header.

The Microsoft C compiler comes with a nifty little program called exemod that does just that—messes around with the .EXE file header to change the default run-time size of the program. Unfortunately the Microsoft version is needlessly difficult to use (requiring you to specify stack sizes in hex bytes and heap sizes in decimal paragraphs), and, of course, if you don't have the compiler, you don't have exemod either. An easier-to-use version of exemod is in Listing One (page 100).

The program (called exe) can be used in one of three ways (shown in Table 1, page 107, along with a sample output). If no command-line switches are present, then exe just prints the contents of the header. I'll look at this header in greater depth in a moment. The -m flag is used to change the default data area size (the combined sizes of the stack, heap, and bss areas). If N is too small (less than the combined bss and stack sizes), then it's rounded up to the minimum. You can

use -m1 to get the smallest possible run-time image, though the image will grow larger if the program ever calls malloc(). The -sN switch increases or decreases the stack size to N bytes. If necessary, the run-time image will be made larger to accommodate a larger stack. The image isn't made smaller when you reduce the stack size. You can run exe twice, however, reducing the stack size the first time and then reducing the total file size the second time. For example:

exe - s1024 file.exe exe - m1 file.exe

reduces a file's stack to 1,024 bytes and then eliminates the space allocated to the heap. Be careful about reducing the stack of a Microsoft-compiled program to less than 1K—I always seem to get a stack overflow error message when I do this. God knows what all that stack space is used for—my own part of the program isn't using it.

Note that the largest N that can be associated with either switch is 65,535. The only reason for this limitation is that I've used *getargs()* to process command-line arguments and *getargs()* can't handle *long-size* arguments very easily. If you want larger images, replace the *getargs()* call on line 68 with your own command-line processing routine.

The .EXE file header is defined by the structure on lines 22—39, reproduced in Code Example 1, page 107. The *signature* is a unique number used to identify this file as an .EXE file. The *file\_size*, header\_size, and image\_len fields are used to determine the size of the load module (the combined text and initialized data areas). In particular, the load image requires

 $\begin{array}{c} ((file\_size * 512) - (header\_size * 16)) \\ + \ image\_len \end{array}$ 

Dr. Dobb's	Journal	of Software Tools	

numbers at right for more info.

Name		
Title		
Company		
City/State/Zip January 1987 #123	Expiration Date:	April 30, 198
Please circle one letter in ea	nch category:	
I. My work is performed: A. for in-house use only. B. for other companies. C. for end users/retailers. D. in none of the above areas II. My primary job function: A. Software Project Mgmt/Sp. B. Hardware Project Mgmt/Sp. C. Computer Consultant D. Corporate Consultant E. Other III. My company department per A. software development. B. computer system integrati C. computer manufacturing. D. computer consulting. E. computer research F. none of the above. IV. This inquiry is for: A. a purchase within 1 month B. a purchase within 1 to 6 m C. product information only.	A. 10,000 or m  A. 10,000 or m  B. 500 to 9,99  C. 100 to 499  D. 10 to 99  E. less than 10  VII. On average, I computers: A. more than c on. B. once per de C. once per w D. less than on  VIII. In my job func A. design soft B. design soft B. design soft C. write code.	on-maker acommend e buter Users at my Jobsit nore 9  advise others about once per day. ay. eek. nce per week. ction, I: ware and/or write code. ware and/or write code.

at the price of \$29.97

# TAKE THIS CARD WITH YOU **AS YOU READ THROUGH** THIS ISSUE OF DR. DOBB'S.



**BUSINESS REPLY MAIL** 

First Class Permit #217, Clinton, Iowa

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of

P.O. Box 2157 Clinton, Iowa 52735-2157

NO POSTAGE **NECESSARY** IF MAILED IN THE UNITED STATES





FIRST CLASS PERMIT #217, CLINTON, IOWA

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's Journal of

P.O. Box 2157 Clinton, Iowa 52735-2157

NO POSTAGE **NECESSARY** IF MAILED IN THE UNITED STATES



# TAKE THIS CARD WITH YOU **AS YOU READ THROUGH** THIS ISSUE OF DR. DOBB'S.

	1	2	3	4	5	6	7	8	9	10
	11	12	13	14	15	16	17	18	19	20
	21	22	23	24	25	26	27	28	29	30
	31	32	33	34	35	36	37	38	39	40
	41	42	43	44	45	46	47	48	49	50
	51	52	53	54	55	56	57	58	59	60
8	61	62	63	64	65	66	67	68	69	70
3	71	72	73	74	75	76	77	78	79	80
	81	82	83	84	85	86	87	88	89	90
3	91	92	93	94	95	96	97	98	99	100
	101	102	103	104	105	106	107	108	109	110
	111	112	113	114	115	116	117	118	119	120
	121	122	123	124	125	126	127	128	129	130
	131	132	133	134	135	136	137	138	139	140
	141	142	143	144	145	146	147	148	149	150
	151	152	153	154	155	156	157	158	159	160
2	161	162	163	164	165	166	167	168	169	170
	171	172	173	174	175	176	177	178	179	180
	181	182	183	184	185	186	187	188	189	190
2000000	191	192	193	194	195	196	197	198	199	200
	201	202	203	204	205	206	207	208	209	210
	211	212		214				218		220
	221	222	223	224	225	226	227		229	230
	231	232			235				239	240
	241	242	243	244		246	247	248	249	250
	251	252			255			258	259	260
	261	262	263	264	265	266	267	268	269	270
	271	272			275				279	280
	281	282	283	284	285	286	287		289	290
	291	292				296		298		300
	301	302	303	304	305	306	307	308	309	310
	311	312	313		315					320
	321	322	323		325	326	327		329	330
	331	332			335					
	341	342 352	343 353	344 354	345 355			348 358	349 359	350
	351 361	362	363	364	365	366			369	370
	371								379	380
	381	382	383	384		386			389	390
	391	392				396		398	399	999
	1000000000									
					rt a 1		onth	sub	scrip	TIOI

Dr. Dobb's Journal of Softv	ware	Tool	S
-----------------------------	------	------	---

January 1987 #123	<b>Expiration Date:</b>	April 30, 1987
City/State/Zip		
Address		
Company	Phone	
Title	P	
Name		

#### Please circle one letter in each category:

- I. My work is performed:
  - A. for in-house use only. B. for other companies.
  - C. for end users/retailers.
  - D. in none of the above areas.
- II. My primary job function:
- A. Software Project Mgmt/Spvr
  - B. Hardware Project Mgmt/Spvr C. Computer Consultant
  - D. Corporate Consultant
- E. Other

#### III. My company department performs:

- A. software development.
- B. computer system integration.C. computer manufacturing.
- D. computer consulting.
- E. computer research F. none of the above.
- IV. This inquiry is for:
  - A. a purchase within 1 month.

    B. a purchase within 1 to 6 months.

  - C. product information only.

#### April 30, 1987

#### V. Corporate Purchase Authority:

- A. Final Decision-maker B. Approve/Recommend
- C. No Influence

#### VI. Personal Computer Users at my Jobsite:

- A. 10,000 or more
- B. 500 to 9,999 C. 100 to 499
- D. 10 to 99
- E. less than 10

#### VII. On average, I advise others about

- computers:
- A. more than once per day.
- B. once per day.
- C. once per week
- D. less than once per week.

#### VIII. In my job function, I:

- A. design software and/or write code.
- B. design software.
- C. write code
- D. don't design software or write code.

A Reader Service number appears on each advertisement. Circle the corresponding numbers at left for more info

# C Programmers! **High-Speed Database tames** complex Capplications

"db\_VISTA™ lets you easily build complex databases with many interconnected record types..." Dave Schmitt, President, Lattice, Inc.

igh-Speed data retrieval and access. just two benefits of using RAIMA's network model DBMS, db\_VISTA. Combine these design benefits with those of C-speed, portability, efficiency, and you begin to understand db\_VISTA's real measure... performance.

#### **Independent Benchmark proves** High-Speed model 2.76 times faster

An independent developer benchmarked db\_VISTA against a leading competitor. Eleven key retrieval tests were executed with sequentially and randomly created key files.

\*Result of 11 Key Retrieval Tests

db\_VISTA :671.24 seconds Leading Competitor :1,856.43 seconds

db\_VISTA's high-speed network database model lets you precisely define relationships to minimize redundant data. Only those functions necessary for operation are incorporated into the run time program.

## Portable DBMS Applications with db\_VISTA

For maximum application portability, every line of db\_VISTA's code is written in C and complete source code is available. db\_VISTA operates on most popular computers with several operating systems supported. So whether you write applications for micros, minis, or main-frames...db\_VISTA is for you.

#### How db\_VISTA works...

Design your database and compile your schema file with the database definition schema file with the database definition language processor. Develop application programs, making calls to db\_VISTA's C functions. Edit and review your database using the Interactive Database Access utility. Compile and link your C program with the db\_VISTA runtime library, and your application is ready to run.

#### Multi-user and LAN capability

Information often needs to be shared. db\_VISTA has multi-user capability and supports simultaneous users in either multi-tasking or local area networking environments, allowing the same C applications to run under UNIX, MS-DOS, and

#### db\_QUERY™lets you ask more of your database

\_QUERY is a linkable, royalty-free, SQL-based ad hoc query and report writing facility. It provides a user-friendly relational view of a network-model database. Use it directly or design an interface for the inexperienced to generate powerful queries



High-Speed Programming Tools, Designed for Portability

#### Royalty-Free Run-Time

Whether you're developing applications for a few customers, or for thousands, the price of db\_VISTA or db\_QUERY is the same. If you are currently paying royalties for a competitor's database, consider switching to db\_VISTA and say goodbye to rovalties.

#### **FREE Technical Support** For 60 days

Raima's software includes free telephone support and software updates for 60 days. Technical support personnel are available to answer questions about our software or

#### 30-Day Money-Back Guarantee

Try db\_VISTA for 30 days and if not fully satisfied, return it for a full refund.

#### Order Schedule

	db	VISTA	db_	QUER
☐ Single-user	\$	195	\$	195
☐ Single-user w/Source	\$	495	\$	495
☐ Multi-user	\$	495	\$	495
☐ Multi-user w/Source	\$	990	\$	990
□ VAX Multi-user	\$	990	\$	990
☐ VAX Multi-user w/Source	\$	1980	\$1	980

Not Copy Protected

#### Call Toll-Free Today!

Order Line . . . . . 1-800-327-2462 Information Line . 1-206-828-4636





#### Read what others say...

"If you are looking for a sophisticated C programmer's database, db\_VISTA is it. In either a single or multi-user environment, db\_VISTA lets you easily build complex databases with many interconnected record types. The multi-user implementation handles data efficiently with a LAN, and Raima's customer support and documentation are excellent. Source code availability and a royalty-free run-time is a big plus."

Dave Schmitt, President

Lattice, Inc.

"My team has developed a sophisticated PC-based electronic mail application for resale to HP customers. db\_VISTA has proved to be an all-round high performer in terms of fast execution, flexibility and portability, and has undoubtedly saved us

much time and development effort."

John Adelus, Hewlett-Packard Ltd. Office Productivity Division

"On the whole, I have found db\_VISTA easy to use, very fast with a key find, and powerful enough for any DBMS use I can imagine on a microcomputer.

Michael Wilson, Computer Language

#### db VISTA Version 2.2

#### **Database Record and File Sizes**

- · Maximum record length limited only by accessible RAM
- Maximum records per file is 16,777,215 · No limit on number of records or set
- · Maximum file size limited only by
- available disk storage

  Maximum of 255 index and data files
  - **Keys and Sets**
- Key length maximum 246 bytes
- · No limit on maximum number of key fields per record—any or all fields may be keys with the option of making each key unique or duplicate
- No limit on maximum number of fields per record, sets per database, or sort fields per set
- No limit on maximum number of member record types per set

- Operating System & Compiler Support Operating systems MS-DOS, PC-DOS, UNIX, XENIX, SCO XENIX, UNOS, ULTRIX, VMS
- C compilers: Lattice, Microsoft, DeSmet, Aztec, Computer Innovations, XENIX and UNIX

#### Features

- · Multi-user support allows flexibility to run on local area networks
- File structure is based on the B-tree indexing method and the network database model
- Run-time size, variable—will run in as little as 64K, recommended RAM size is 256K
- Transaction processing assures multi-user database consistency
- File locking support provides read and write locks on shared databases
   SQL-based db\_QUERY is linkable
   File transfer utilities included for ASCII, dBASE optional

#### Utilities

- Database definition language processor Interactive database access utility
- Database consistency check utility
- Database initialization utility

- Multi-user file locks clear utility
   Key file build utility
   Data field alignment check utility
   Database dictionary print utility
   Key file dump utility
- Key file dump utility
- · ASCII file import and export utility

\*The benchmark procedure was adapted from "Benchmarking Database Systems: A Systematic Approach" by Bitton, DeWitt and Turbyfill, December 1983.

3055-112th Avenue N.E. Bellevue, WA 98004 USA (206) 828-4636 Telex: 9103330300

Order Toll-Free 1 (800) 327-2462 C CHEST (continued from page 104)

bytes. In Table 1, this comes to

((22\*512)-(32\*16)) + 124

or 10,876 bytes.

Several of the fields are used for patching up a few instructions that the linker can't patch. There are num\_reloc of these items (three in exe.exe) organized as a linked list with the first node in the list at offset first\_reloc from the beginning of the load module.<sup>1</sup>

The bss\_min and bss\_max fields are used to allocate the combined heap, stack, and bss space. The initialized data, because it's stored on the disk, is considered to be part of the load module, so its size isn't duplicated here. Bss\_min is the minimum amount of required memory in paragraphs (16-byte chunks). It's the combined bss and stack sizes. Bss\_max determines the maximum amount of allocated memory (also in paragraphs), so if it's larger than bss\_min, the difference between the two numbers is the amount of heap that can be allocated before DOS has to be called. The default values of bss\_min and bss\_max for exe.exe are shown in Table 1 (196 and 65,535, respectively). This means that the program requires a minimum of 196 paragraphs (3,136 bytes) for the combined bss/stack area and will use all the rest of memory for the heap. The smallest-possible image can be created by setting bss\_max to bss\_min.

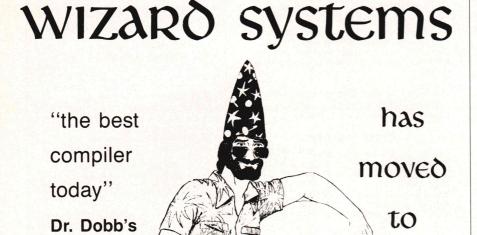
The <code>init\_sp</code> and <code>stack\_disp</code> fields are used to set up the stack size. <code>Init\_sp</code> is both the stack size and the initial value of the <code>SP</code> register (the <code>SS</code> register points at the bottom of the stack area). <code>Stack\_disp</code> is used to locate the bottom of the stack. It is added to the initial contents of the <code>CS</code> register to initialize the <code>SS</code> register when <code>DOS</code> loads the program. Note that the stack size is included in the <code>bss\_min</code> and <code>bss\_max</code> figures, so these will have to be modified if <code>init\_sp</code> is made larger.

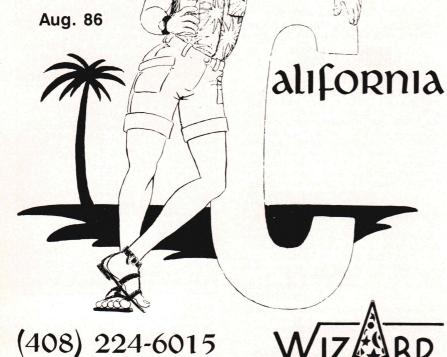
All these transformations are done by the code in Listing One. The file is opened on line 74, the .EXE header is read on line 77, the <code>bss\_min</code> and <code>bss\_max</code> fields are modified on lines 81–97, and the stack variables are modified on lines 99–122. The .EXE header is written back out on both lines 95–96 and 120–121 (you have to seek back to the start of the file before writing). Finally, the header contents are printed by <code>print\_hdr()</code>, called on line 124.

The fatal\_err() subroutine is given in Listing Two, page 102. It is used just like printf() is used. It writes a message to stderr and then exits to the operating system. Note that I've used the ANSI (as compared to Unix) conventions for subroutines with a variable number of arguments. Va\_list and va\_start are macros defined in stdarg.h, supplied with the compiler. If your compiler doesn't support these, substitute calls to fprintf() and then exit() for the fatal\_err() calls. I'll talk more about subroutines with a variable number of arguments in a future column.

#### Erratum

The nonrecursive binary-tree traversal routine presented in July has a serious bug in the algorithm. It couldn't handle the case of a leaf that had a right, but no left, descendant. Listing Three, page 103, is another version of the routine that seems to work correctly. The basic process is





Only \$450





17645 Via Sereno Monte Sereno, CA 95030 still the same (descend the tree reversing pointers so you can go back up again, setting a tag bit just before going right), but the code has been shuffled around a bit. Note that in this version I'm keeping a tag field in the structure rather than setting the high bit of the first character of the key string. Look back in the July C Chest if you need a more detailed explanation of what's going on.

#### **Availability**

All the code from this month is available on CompuServe in DL1 (type

ddjforum). The *getargs()* subroutine, used but not printed this month, was originally published in the May 1985 C Chest. The version used here has a fifth argument not present in the original version. If you're using the earlier version, just omit the extra argument. The current version of *getargs* is available both on CompuServe and as part of the /util program disk distributed by *DDJ* (see ad, page 124).

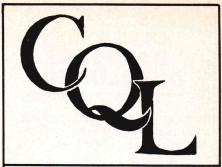
All the source code for articles in this issue is available on a single disk. To order, send \$14.95 to *Dr. Dobb's* 

```
exe -mN file
                    Modify the maximum memory used to N bytes. If this number is
                    smaller than the combined bss and stack sizes, then it's rounded up.
                    Use -m1 for the smallest possible load module. The maximum value
                    of N is 65,535.
                    Modify the stack size to be N bytes.
exe -sN file
exe file
                    Just print out the contents of the .EXE header. The command line
                    exe exe.exe generated the following:
   23117 (0x5a4d):
                       Signature (marks this as a valid .EXE file)
      124 (0x007c):
                       Length of image mod 512
       22 (0x0016):
                       File size (512-byte blocks) including header
        3 (0x0003):
                       Number of relocation table entries
       32 (0x0020):
                       Size of the header (paragraphs) = 512 bytes
      196 (0x00c4):
                       Min. memory above program (paragraphs) = 3.136 bytes
   65535 (0xffff):
                       Max. memory above program (paragraphs) = 1,048,560 bytes
      715 (0x02cb):
                       Displacement (paragraphs) of stack within load module
     2048 (0x0800):
                       Initial value of the SP register (= the stack size)
 -14653 (0xc6c3):
                       Checksum for file
     2008 (0x07d8):
                       Initial value of the PC (IP) register
        0 (0x0000):
                       Displacement (paragraphs) of code seg. within load module
                       Displacement (bytes) to first relocation item in module
       30 (0x001e):
        0 (0x0000):
                       Overlay number
```

#### Table 1: Using exe

```
typedef unsigned short word; /* 2-byte unsigned number */
typedef struct
  word
         signature:
                        /* Length of load module image % 512
  word
         image_len:
  word
         file_size;
                        /* File size in 512-byte units
                                                               */
  word
         num_reloc;
                        /* Number of relocation table items
                                                               */
  word
         header_size; /* Size of the header in paragraphs
                                                               */
  word
         bss_min;
                        /* min size of data area above program
                                                               */
  word
         bss_max;
                        /* max size of data area above program
                                                               */
  word
         stack_disp;
                        /* displacement in para. of stack seg.
                                                               */
  word
         init_sp;
                        /* Initial SP register contents
                                                               */
  word
         checksum;
                        /* Checksum for file
                                                               */
         init_ip;
  word
                        /* Initial IP register contents (PC)
  word
         code_disp;
                        /* displacement in para. to code seg.
         first_reloc;
  word
                        /* displacement (bytes) to 1st re-
                            loc item
  word
         overlay;
                        /* overlay number.
                                                               */
EXE_HEADER;
```

Code Example 1: The .EXE file header



SQL Compatible Query System adaptable to any operating environment.

**CQL Query System.** A subset of the Structured English Query Language (SEQUEL, or SQL) developed by IBM. Linked files, stored views, and nested queries result in a complete query capability. File system interaction isolated in an interface module. Extensive documentation guides user development of interfaces to other record oriented file handlers.

#### Portable Application Support System

Portable Windowing System. Hardware independent windowing system with borders, attributes, horizontal and vertical scrolling. User can construct interface file for any hardware. Interfaces provided for PC/XT/AT (screen memory interface and BIOS only interface), MS-DOS generic (using ANSI.SYS), Xenix (both with and without using the curses interface), and C-library (no attributes).

Screen I/O, Report, and Form Generation Systems. Field level interface between application programs, the Query System, and the file system. Complete input/output formatting and control, automatic scrolling on screens and automatic pagination on forms, process intervention points. Seven field types: 8-bit unsigned binary, 16 bit unsigned binary, 32 bit signed binary, monetary (based on 32 bit binary), string, and date.

#### \$395.00

C Interpreter. Run the interpreter on any hardware and on any operating system. Develops true intermediate code, allowing full C features in an interpreter. User configurable interface to compiler library allows linkage with compiled routines.

HARDWARE AND FILE SYSTEM INDEPENDENT

## Kurtzberg Gomputer Systems

41-19 BELL BLVD. Bayside, N.Y. 11361

VISA/Master Charge accepted **(718) 229-4540** 

\*C-tree is a trademark of FairCom

IBM, SEQUEL, PC, XT, AT are trademarks of IBM Corp. MS-DOS and Xenix are trademarks of Microsoft Corp.

CQL and the CQL Logo are trademarks of Kurtzberg Computer Systems.

Circle no. 294 on reader service card.

### ATTENTION DATABASE **DEVELOPERS**

If you have the need for relational database management within your application programs

#### TURBO-DB™

is the programmer's tool you have been waiting for!

TURBO-DB offers a number of features for the software developer:

SMALL AND LARGE DATABASE MANAGEMENT

AUNIQUE

**PROGRAMMATIC** INTERFACE FOR **CUSTOM APPLICATION** DEVELOPMENT

IN ASM, 'C', FORTRAN AND PASCAL

QUEL-TYPE **QUERY LANGUAGE** 

INTEGRATED DATABASE **ADMINISTRATION** 

DATABASE RECOVERY AND BACKUP UTILITIES

WRITTEN ENTIRELY IN 'C'-UNIX COMPATIBLE

For the IBM PC/XT/AT or compatible computers (256K RAM, DOS 2.0+, diskette or hard disk)

#### Software Developer Package I

- Interactive Query Writer
- Database Administration Database Backup and
- Recovery Utilities
- **User Tutorial**
- User Reference Manual

\$145

#### Software Developer Package II

- Software Developer Package I
- 'C' Object Code Interface
- Programming Interface Reference Manual

\$395

#### UPLAND SOFTWARE CO.

P.O. BOX 3136 • REDWOOD CITY, CA 94064 (415) 876-7636

MasterCard, Visa, Checks Accepted

C CHEST (continued from page 107)

Journal, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number and disk format (MS-DOS, Macin-

#### Note

tosh, Kaypro).

1. For more information about how the relocations are processed consult Chapter 10 of IBM Corp.'s DOS Technical Reference (Boca Raton, Fla.: IBM Corp., 1985). Much better descriptions of relocatable object module formats in general are in Steven Armbrust and Ted Forgeron's ".OBJ Lessons," PC Tech Journal 3:10 (October 1985), 63-81, and Intel Corp.'s 8086 Relocatable Object Module Formats (Santa Clara, Calif.: Intel Corp., 1981), order number 121748-001.

DDJ

(Listings begin on page 100.)

Vote for your favorite feature/article. Circle Reader Service No. 6.

#### **Flotsam and Jetsam**

#### **Memory Organization** in a C Program

Most C programs are segmented into five parts: the code area (called the text segment); the initialized data space (called the data segment); the uninitialized data space (or bss segment; the stack space (or stack segment); and the area of memory used by malloc(), called the heap.

The stack is used for subroutine calling, in the normal way, but it's also used to store local automatic variables. When a subroutine is called, it subtracts a constant from the stack pointer to make room on the stack for its own local variables and then accesses those variables indirectly through either the stack pointer or a special register called the frame pointer.

Variables at fixed addresses (globals and local statics) are in either the data or bss segments, depending on whether they are initialized by your program. Variables can be initialized explicitly (with an equal sign as part of the declaration) or implicitly. An example of the latter is a string constant, such as a format string passed to a printf() call. Here the compiler automatically allocates and implicitly initializes an area of memory to hold the string, and that memory is put into the data segment (rather than the bss segment).

Usually, both the text and data segments (code and initialized data areas) are stored on the disk together. When the program is loaded, the variables in the data segment are loaded directly into their correct place in memory. There's no code generated to initialize static data; the data that is read in from the disk has the correct initial value. This explains why a static local variable has its initial value the first time a subroutine is called but on subsequent calls the variable contains the same value that it had at the end of the previous call. It's read in having the initial value, but once you change it, it stays changed.

The remaining three memory areas (bss, stack, and heap) are created as part of the loading process. After the loader has transferred the text and data segments from the disk into memory, it allocates enough additional memory above the data segment to contain the other three segments. The loader usually sets up the stack pointer to point into the stack. The program itself (or more correctly, the root or start-up module) initializes the entire bss segment with 0s and then calls your main() subroutine.

The usual order of segments, going from low to high memory, is:

text (code) data (initialized data) (uninitialized data) bss (local variables) stack heap (used by malloc())

However, the stack and heap are often reversed. The text and data areas are always adjacent because they're read from the disk as a single unit.

# C Programmers: We support every product in this ad & 700 others. Try any product here with a full 31-day money-back guarantee.

#### Add Multitasking Communications Multi-Comm

Add background communications to an existing system or write code for drawing from multiple devices,

networking, or multiple users. Multi-Comm is fully interrupt driven, ROMable, and handles rates up to 9600 baud. It supports multiple tasks, ports, and users . . . all within your program. Designed to work with Multi-C, Multi-Comm has all the functions you need for error-free ASCII or binary data transfer.

XMODEM protocol handler allows vou to transfer data between your program and just about any other computer. Modem control functions allow you to place or answer calls

from foreground or background.

Multi-C lets you create, manage. and communicate among tasks with little overhead.

All Multi-Comm hardware dependent modules are supplied in source form (primarily in C). No royalties. MS C, Lattice 2.15 or 3.0. C86.

Multi-Comm with Multi-C MSDOS \$298 Multi-C Only MSDOS \$149

Circle no. 301 on reader service card.

#### Fastest C **Development on Earth:** Instant C version 2.0

Instant C's NEW version 2.0 gives you immediate (2 secs.) compilation and execution of large (15,000 - line) programs, and the ability to link in external (commercial or your own) libraries in an interactive. Lattice 3.0 or Microsoft 3.0 compatible, interpreter-like environment with an integrated full screen editor and source level debugger.
You'll get full K&R standard C fast (33 second sieve) execution

speed, and debugging with source code animation, single-stepping, backtracing, and unlimited conditional breakpoints.

Instant C now supports multiple screens and graphic devices, run-time checking of pointer and arrary references, and includes a new manual, expanded tutorial and reference section, and complete library source.

#### 617-653-6194 MSDOS \$399

#### Rational

Systems, Inc.

Circle no. 302 on reader service card.

#### A NEW C STANDARD FOR SCREENS AND WINDOWS: C-SCAPE

Setting a new standard for screen generation, C-scape turns your Dan Bricklin Demo Program screens into C code instantly. You can capture existing screens from 1-2-3, Turbo, or that old BASIC diehard and convert them to C in seconds. C-scape can save you immense effort and reduce errors for both new program development and language conversion projects.

C-scape is a combination screen generator and library of input/output functions that provides an advanced and powerful ability to create different types of menus, input fields, help screens, and text with unprecedented speed and flexibility. Tiled, pull-down, and pop-up windows of virtually any depth (limited by RAM) are a key feature, along with scrolling, full color and type support, and individual key or field validation.

Because C-scape is based on C's printf statement, you can embed the commands for screen positioning and field definition right inside your format string. This helps you produce clear, readable code, which is easier to maintain and change.

Since full source code is provided, the standard library routines can be tailored to meet your exact screen layout and keystroke handling requirements.

All C programmers will benefit from C-scape's readable, intuitive syntax, based on an extension of C's printf function. Beginners will learn by studying code generated from captured screens. Advanced programmers will enjoy C-scape's ease of maintenance. Power programmers will appreciate the free source code provided at no additional cost upon registration.

Oakland Group, Inc. features free updates, an on-line bulletin board for users, and toll-free technical support at 800-233-3733 (800-BEE-FREE)

Escape the pitfalls of coding from scratch, and free up your time for creativity and productivity. Buy C-scape now and take advantage of the 31-day review period: satisfaction guaranteed or your money back. No royalties. No license fee. Lattice 3.0, Microsoft 3.0 & 4.0.

Oakland Group, Inc.

**PCDOS \$149** 

Circle no. 303 on reader service card.

#### **BRIEF Makes Editing C Programs a Breeze**

No other editor comes close to making your life easier. BRIEF, The Programmer's Editor, is tailored to address the needs of C programmers — your needs. Take a look at the BUILT-IN features below. They are just part of the reason why 1000s of C programmers already rely on BRIEF

AutoCompile - While in BRIEF you can compile with MS C, Lattice, or several other compilers (we'll even help you support any compiler through our 800-line tech support). You save over 20 seconds each compile; and you can automate as with a CC.EXE or MAKE.

AutoIndent - Save keystrokes and increase style consistency. Use the editor's default indentations or modify them to your taste.

Template Editing - Get the full structure of a programming construct on-screen and move the cursor from one "fill-in-the-blanks" location to the

Error to Error Tracking - Use "next error" to move to the next appropriate line. Display error messages in a separate window - even when you add or delete code, BRIEF knows where to go.

MultiWindow Editing - Keep different parts of the same file in horizontal or vertical windows. Put your header file in one, main function in another, current function in a third. Any size, any number.

Macro Language - Completely readable and programmable, the macro language will be second nature for an experienced C programmer. Feel free to modify the macro source code included with BRIEF.

And remember, BRIEF is a full featured editor that can be used with any language. User surveys indicate that even beginning programmers become productive with BRIEF in less than 30 minutes. Call The Programmer's Shop and ask about UNDO (not undelete), Unlimited File Size, Tiled & Pop-Up Windows . . ., or ask for a detailed product description that will tell you why BRIEF can't be beat.

PCDOS \$195 **PCDOS \$195** 

800-821-2492 Circle no. 305 on reader service card.

#### FAST, Easy-to-Use Graphics, Royalty-Free: **Essential Graphics**

Draw fast dots, lines, circles, arcs, rectangles, and box fills. Draw a bar or exploded pie chart or a shaded line graph with one function call. Use the font and clip-art manipulation routines with the 10 fonts included (up to 8 simultaneously, or choose from over 500 other fonts and clip-art sets available

Essential Graphics provides fast animation and graphic windowing using GET and PUT, and generates compact code. Demonstration programs and comprehensive manual included.

Supports IBM Color, EGA, and Hercules cards, Epson and Oki printers. Lattice, Aztec, C86, Desmet, MS C, others. No royalties
201-762-6965

**PCDOS \$219** 

Circle no. 304 on reader service card.

#### C DYNAMO! WINDOWING: Full C Source, No Royalties

Power Windows and C Function Library

Power Windows covers all the bases: overlays, borders, 1-2-3 style or pop-up menus/help windows, zap instantly on/off screen, status lines, horizontal/vertical scrolling, color control or highlighting, word-wrap, files to windows, keyboard to windows. Powerful, easy to use, integrated error messages, thorough documentation. Supports IBM monochrome or color. MSDOS Only \$119

C Function Library - includes 325 fundamental functions with readable source and thorough documentation.

MSDOS Only \$119 No matter what you have, you need these. Best value available. Highly recommended! 713.468.4412 713-468-4412

Entelekon

Circle no. 306 on reader service card.

## Call for Your FREE C Programmer's Directory

#### **HOURS**

8:30 AM - 8:00 PM EST.

800-421-8006

THE PROGRAMMER'S SHOP™ 128-D Rockland Street, Hanover, MA 02339 Mass: 800-442-8070 or 617-826-7531 8/86

"I would like to mention that I appreciate the way that the Programmer's Shop does business. It is indeed refreshing to be able to call and get answers that you can trust in, to questions on various products.

Donald E. Winters MIS Software Development Inc.

## STRUCTURED PROGRAMMING

#### **Naming Names**

orth has been designed by programmers who were using it, and so Forth's design is responsive to programmers' needs in small ways as well as large. Other languages don't seem to be quite so programmerfriendly. For example, I was surprised to read in an article about Modula-2 (by this column's own Namir Shammas) the complaint that the underscore character was not allowed in names. How did Modula-2's designer conclude that programmers are helped by disallowing some characters in names? In my heart of hearts, I suspect that the rule was for the benefit of the compiler writers, not the compiler users, and exemplifies fitting the task to the program rather than the other way around. This type of programming focuses on what is easy now (for the program writer), not what is easy over the life of the program (for the program users).

Service workers must always fight the tilt toward serving themselves before the clients of their profession. College administrators who bemoan the loss of serenity when students return to the campus, shelf stockers whose tempers flare when customers disorganize displays by buying items from them, and programmers who have had it up to here with figuring how to help the endlessly confused user—all should remind themselves of the point of the enterprise.

Programmers using a particular language pray that its developers

#### by Michael Ham

kept in mind that they were writing for programmers and made their first objective easing the programmer's life, not their own task. The programmer users hope that the lan-

© 1986 by Michael Ham. All rights reserved



guage developers, weary of considering all the ins and outs of implementation, did not finally throw in the towel and say, "This will be good for you, really. You'll like not being able to use underscores. Anyway, you'll get used to it."

In Forth, any character can be used in a name—well, almost any. Blank doesn't work because it is the name delimiter, and carriage return doesn't work because it marks the end of the line. Generally speaking, however, Forth is not picky about the characters you want to use.

Some standard usages have evolved in which some characters represent a class of tasks. A word beginning with a period normally displays information: .DATE, for example, can be assumed to display the date, .NAME a name, .S the stack (abbreviated because so often used), .FILENAME a file name, and so on. The greater-than symbol is often used for 'to''—to indicate movement (>R puts a character from the data stack onto the return stack, R> takes it off the return stack and returns it to the data stack) or transformation (S>D converts a single-precision number to a double-precision equivalent, >JULIAN converts a date to a Julian date). The symbol? denotes a Boolean flag, and in a program in which names are carefully chosen, its meaning is obvious. For example, *STOP?* would leave a flag true, meaning stop, and *?STOP* would consume a flag true, causing a stop.

Code Example 1 below, shows a tiny tool *YES?* that collects a yes/anything response and leaves a true flag if the user answers yes. The word suggests a yes response (hence the name) by displaying *Y* as the default answer. *YES?* uses *CAP* to capitalize any lowercase input before checking whether it was a *Y*.

Some of these name patterns come from conventions, but conventions are more successful when they recognize and reinforce usage than when they attempt to create it. Rushing the process or trying to fence it in with rules does not lead to better results more quickly but merely frustrates and confuses the evolutionary movement. Forth gives the programmer complete freedom in naming, and the conventions for naming emerge gradually.

Other languages give the compiler writer the authority to decide the sorts of names that would be good for programmers (or, possibly, good for compiler writers) with the result that some characters fall beyond the pale that pens the programmer. "You want underscores in the name? That sort of thing isn't done in Modula-2. Don't be perverse."—the mark of bluestockings, ready to ease their life by adding difficulties to yours. Fight back. Use Forth.

Some programming languages build in conventions through tactics such as precedence rules. One popu-

- : CAP ( c-c ) DUP 96 ) OVER 123 ( AND IF BL THEN ;
- : YES? ( f ) ASCII Y EMIT 8 EMIT ( backspace )

  KEY CAP DUP ASCII Y = SWAP 13 = ( cr? ) OR DUP

  IF ASCII Y ELSE ASCII N THEN EMIT SPACE;

Code Example 1: Two tiny tools

lar language has upward of 20 precedence rules. It's too many. The doctor in John Barth's novel *End of the Road* (New York: Avon Books, 1964) suggests to Jacob Horner that three will suffice: "'If the alternatives are side by side, choose the one on the left; if they're consecutive in time, choose the earlier. If neither of these applies, choose the alternative whose name begins with the earlier letter of the alphabet. These are the principles of Sinistrality, Antecedence, and Alphabetical Priority.'"

Of course, it is always legitimate to propose rules. Ideas can be stimulated through discussion, but their acceptance should ultimately be based upon experience, not fiat. To demonstrate that I am willing to entertain rule proposals, I offer the following suggestions for naming conventions for the arithmetic operators.

The names of the arithmetic operators are perhaps inescapably pedestrian. Forth requires a variety of names because data are not typed and thus the operators are. Operators come in several flavors: single precision, double precision, quad precision, and mixed precision (operations in which the two operands are of different precision, typically one being single precision and the other double precision). Happily, all the mixed-precision operators can be defined to take the lesser precision operand on top of the stack, the greater precision second on the stack.

To simplify the discussion, let's follow FORTH Inc.'s lead and call single-precision numbers *singles* and double-precision numbers *doubles*. There are no mixed-precision numbers, of course, only mixed-precision operations (with a single and a double or a double and a quad as arguments).

The precision of the result of an operation is another question. Normally sums and differences are assumed to have the same precision as the operands that produced them, though that is not logically necessary: a sum of two singles could, for example, be a double. In multiplication you more commonly will want to allow the result to be of a higher precision than the factors (the product of two singles

being a double, for example). And with division you might be content for the quotient to drop back a notch: double divided by single with the quotient a single. Note, however, that it seems best for the remainder to be accepted as a double even if the quotient is taken as a single.

The sign lurks as the high bit of the binary representation of the number, but unsigned numbers use that bit in its numeric meaning. Addition and subtraction do not need to distinguish—the programmer can choose how to interpret the high bit when the result is displayed. But in other operations it can make a difference: is it 10 compared to 65,535 (10 is less) or 10 compared to -1 (10 is greater)? If an operation treats the high bit as number rather than as sign, the operation is called unsigned.

Ideally, the Forth names for the operators could offer the programmer some reliable signposts through this maze of options: single, mixed, double, unsigned, signed, incoming, outgoing. The current crop of names was not designed to offer this kind of

# 80386

#### SOFTWARE DEVELOPMENT TOOLS

The Phar Lap 80386 Software Development Series:

**386 ASM/LINK** by Phar Lap (MS-DOS®) \$495 Full-featured macro assembler and linker. Includes a debugger and 32-bit protected mode runtime environment.

80386 High-C™ by MetaWare (MS-DOS®) \$895

**80386 Professional Pascal™** (MS-DOS®) \$895 by MetaWare

UNIX™ and VAX/VMS® cross tools (call)

The wait for professional software development tools for the 80386 is over! Whether you are upgrading an existing IBM PC application to the '386, moving a mainframe application down to a PC, or writing the next PC best-seller, the Phar Lap 80386 Software Development Series is for you. It is an integrated line of products which provides everything you'll need to create and *run* 80386 protected mode applications under MS-DOS.

(617) 661-1510

Phar Lap Software, Inc. "The 80386 Software Experts"

60 Aberdeen Ave.

Cambridge, MA 02138

Circle no. 343 on reader service card.

# 99<sup>44</sup><sub>100</sub> 0 UNIX EQUIVALENT Power Tools

TOOL BOX 1	\$25
TOOL BOX 2	\$30
SYSMAKE	\$25

#### For MSDOS and CP/M

I/O redirection • wildcard expansion exclusions • full documentation

## NIRVONICS inc

P.O. Box 5062 Plainfield, NJ 07061

(201) 561-2155

Trademarks — UNIX: Bell Labs, MSDOS: Microsoft, CP/M: Digital Research

Circle no. 331 on reader service card.

## STRUCTURED PROGRAMMING (continued from page 111)

help. An alternative scheme is suggested in Tables 1 and 2, below.

Table 1 contains a list of prefixes for the arithmetic operators, based on the precision of the operands. When both operands are signed, single-precision numbers, the operators are unadorned. Otherwise, the operator names include information that describes the nature of the operands.

Table 2 contains a list of suffixes. Just as the prefix describes the nature of the operands (the input), the suffix describes the nature of the result (the output). Again, a garden-variety op-

erator that produces the natural result (for example, an operation on single-precision numbers that produces a single-precision result or on doubles that produces a double). This convention assumes that the "natural" result for a mixed-precision operation has the higher of the precisions of the two operands. For example, the natural result of a single-double mixed operator is a double, and thus no suffix is used in that case. If a single-double operator produces a single result, then it is named with a suffix S to show that the result is single precision.

These names group double-precision operators under *D* and mixed-precision operators under *M*. The affixes allow you to decipher the special qualities of an arbitrary operator and also make it easy to remember the operation names. Table 3 shows a variety of operator names that test this scheme.

Mixed precision normally is an issue only with the input because the output is normally only one number. The /MOD operators, however, produce two numbers as output: a quotient and a remainder. Would it be reasonable to see mixed precision here, with the quotient being one precision and the remainder another?

In integer division, the max of the dividend and the divisor will be larger than the quotient and the remainder, so if the dividend and divisor are both single precision, both quotient and remainder will be single precision. The single-precision /MOD thus leaves single-precision results, and the issue does not arise.

With D/MOD, however, the situation is different. Dividing a double by a double might well produce a single. The remainder, however, could well be a double. Thus, you might want the operation D/MODM, two doubles producing a mixed result: a single-precision quotient and a double-precision remainder. Mirabile dictu, the single is again on top of the stack, the double beneath. (Perhaps single numbers, weighing less than doubles, naturally float to the top of the stack.) You can use M as a suffix to indicate that the mixed precision is on the output side rather than the input:

*D\*/MODM*—three signed doubles, with quad-precision intermediate re-

**Sign Assumption Prefix Operands** Single-precision signed operands none unsigned operands 11 Double-precision signed operands D DU unsigned operands Quad-precision signed operands 0 unsigned operands QU Mixed-precision Single-double signed operands M unsigned operands MU Double-quad signed operands MD MDU unsigned operands

Table 1: Prefixes for arithmetic operators

Operands	Result	Suffix
Single-precision	single precision	none
图图 · 图 · 图图 · 图图 · 图图 · 图图 · 图图 · 图图 ·	double precision	D
Double-precision	single precision	S
	double precision	none
	quad precision	Q
Mixed-precision		
Single-double	single precision	S
	double precision	none
Double-guad	double precision	D
1 福 縣 等 通	quad precision	none

Table 2: Suffixes for arithmetic operators

	The sine desired and significant feature available as double product
*D	Two signed single-precision factors producing a double product.
U*D	Two unsigned single-precision factors producing a double product.
MU*	Mixed-precision factors (single and double), unsigned, with double prod- uct. By convention, the single-precision factor is on top of the stack, the double under it.
MDU*	Mixed-precision factors (double and quad), unsigned, with quad product.
MU>	Mixed-precision unsigned compare. The sense of the comparison is double > single, both considered as unsigned.
D*/	Factors and result are all doubles, with quad-precision intermediate product (The */ operator always takes the intermediate product to the next higher level of precision. Because operators in the */ family have three operands, it seems best to avoid mixed precision on the input side.)
D*	Factors and product all signed doubles.
M*	Single on top of stack, double beneath, with double-precision product, all signed.
D/	Two signed doubles divided, producing double as quotient.
D/S	Two signed doubles divided, producing signed single as quotient.
M/S	Double divided by single with single quotient, all signed.
D/MOD	Two doubles divided, producing a double quotient and a double remainder.

Table 3: Examples of operator names

sult, producing a mixed-precision result: single quotient on top of stack and double remainder beneath.

Q\*/MODM—three unsigned quads, with octuple-precision intermediate result, producing a quad remainder (second on the stack) and a double quotient (top of stack).

The example  $Q^*/MODM$  is a grotesquerie that you would probably never encounter. It serves merely to illustrate that even novel operations can be deciphered easily with this scheme.

If you accept that all M operators require the single number on top of the stack and the double beneath, you can define the following operators that might be useful in mixedprecision situations. But beware of the syndrome of maniacal completeness, in which you define fistfuls of operators to complete a set of logical possibilities, even if those operators are seldom or never used. I suggest that these words be defined only in applications doing a lot of mixed-precision calculation. In that setting, their descriptive names make the code more readable and thus justify their existence.

 $\begin{array}{ll} \text{MSWAP} \ (\,d\,n-n\,d\,) \\ \text{MOVER} \ (\,d\,n-d\,n\,d\,) \\ \text{MNIP} \ (\,d\,n-n\,) \\ \text{MTUCK} \ (\,d\,n-n\,d\,n\,) \end{array}$ 

Some of the names I have suggested for the arithmetic operators don't match names in the 83 Standard. I see no problem in this; the names I propose could be adopted and the older names kept as synonyms. Forth programmers can be encouraged to shift to the new names by having a tiny speed penalty associated with the older names—for example, by defining the older name as an alias of the newer name. Any speed penalty, however slight, is more than enough to make most Forth programmers switch to the faster name.

#### The Toolbox

This perhaps seems like a lot of attention lavished on names, particularly for a publication whose title includes the phrase "software tools." But names are an important part of a programming tool. In a well-designed hand tool, the grip gets serious attention as the primary ergonomic inter-

face, which plays a major part in determining the effectiveness of the tool. The programmer uses names and verbal constructs to manipulate the power of the computer. If those names and constructs fit well the habits of the mind, the task is done that much more easily.

I have proposed new operators as well as new names. I believe that any Forth package should include as a standard component a complete set of double-precision operators, with quad-precision operators available as an extension for 16-bit Forths. Double precision is often needed when working with large numbers in which round-off errors must be minimized, as in accounting applications. Sometimes (in 16-bit Forths) even double precision does not offer enough range and quad precision must be used.

These are the double-precision operators I believe should be present:

D+ D- D\*\*D D/MOD D\*/MOD D>  $D=D < \\ 2SWAP 2OVER 2DROP D2DROP 2DUP \\ D2DUP 2ROT \\ 2, 2@ 2! \\ 2CONSTANT 2VARIABLE$ 

With these at hand, programmers can easily construct other double-precision operators that might be needed: D/,  $D^*/$ , D0=, D0>, and so forth. I suggest D2DROP instead of 4DROP and D2DUP instead of 4DUP because the former show the intention with more clarity and less mental arithmetic.

MMSForth, published by Miller Microcomputer Services, provides a different (and complete) solution to the need for operators of higher precision. Instead of offering optional quad precision, octuple precision, and so on, MMSForth generalizes the idea of integer precision.

The Utilities option for Version 2.4 of MMSForth includes an optional extension called *N-LEN#*. The *N-LEN#* operators parallel the usual number and stack operators and use the same names except that # is included as an identifier. All the operators (#+, #DUP, #OVER, <##, ##S, #\*/MOD, and so on) work by reference to the value of #PREC, which the user sets.

#PREC specifies the number of cells to be used in the arithmetic operations. Setting #PREC to 1 produces the

normal single-precision operators, and setting \*PREC to 2 produces the double-precision operators. But you can set it to arbitrarily high values to allow integer arithmetic of arbitrary precision.

Do not think these operations are sluggish. Test routines included with the package time the computation of Fibonacci numbers and factorials. I computed and printed the 277th Fibonacci number in 1.08 seconds and the number 46! (46 factorial) in 1.50 seconds (both on an IBM PC with the NEC V20 chip instead of the 8088). The number of (2-byte) cells of precision specified in these test routines was 50 for the Fibonacci test and 540 for the factorial test. For most uses, 540 precision seems more than ample. Allowing users to specify the number of bytes of precision they need is clearly a better solution than hand-tailoring operators of various precision.

It should be noted that Version 2.4 of MMSForth, which runs on the IBM PC and on the Radio Shack Model 4 and equivalents, is a native-mode Forth, incompatible with the normal operating systems on those machines. When MMSForth is used, it monopolizes the machine and its resources.

#### Names, Names, Names

One problem Forth programmers face on large projects or when working as a programming team is the number of names that are generated. John James has called this "the name explosion." Because Forth programs are best written as a collection of useful tools (short definitions with general utility), the situation is particularly acute. Compared to procedural languages, Forth systems have more names (shortness of definition) and it is more important to know them (general utility).

Forth programmers generally agree also on the importance of finding the "right" name. The criteria for rightness vary, with one major division between those who prefer playful names and those of more serious mien. Both parties agree, however, that the best names accurately and immediately convey the idea of the word's function. Both parties prefer short names to long. And both find it difficult to get precisely the right name when names must be assigned continually.

# The C Programmer's Assistant

# C TOOLSET

#### UNIX-like Utilities for Managing C Source Code

No C programmer should be without their assistant—C ToolSet. All of the utility programs are tailored to support the C language, but you can modify them to work with other languages

Source code in standard K&R C is included; and you are welcome to use it with any compiler (UNIX compatible) and operating system you choose.

#### 12 Time Savers

DIFF - Compares text files on a line-by-line basis; use CMP for byte-by-byte. Indispensable for showing changes among versions of a program under development.

GREP - Regular exp. ssion search. Ideal for finding a procedural call or a variable definition amid a large number of header and source files

FCHART - Traces the flow of control between the large modules of a program. PP (C Beautifier) - Formats C program files

so they are easier to read.

CUTIL - A general purpose file filter.

Requires MSDOS and 12K RAM

CCREF - Cross references variables used within a program.

CBC (curly brace checker) - checks for pairing of curly braces, parens, quotes, and comments.

Other utilities include DOCMAKE, ASCII, NOCOM, and PRNT.

Source code to every program is included!

#### Thorough User Support Text and Online

C ToolSet documentation contains descriptions of each program, a listing of program options (if any), and a sample run of the program.

On-line help gives you information on the programs and how to run them. Most of the programs respond to -? on the command line with a list of options.

Call 800-821-2492 to order C ToolSet risk-free for only \$95.



335-D Washington St., Norwell, MA 02061 (617) 659-1571 Ful refund if not sails full refund in an days.

Circle no. 152 on reader service card.

# Feel Constrained By PC BATCH Languages?

You need OPAL, a rich, full-functioned executive shell language for the experienced programmer.

Previously, you've had only limited capabilities with the batch exec languages available on the IBM PC. Now, OPAL frees you from the limitations and drudgery of BATCH, and other primitive executive languages.

OPAL, as an interpretive language, makes it easy to program the control you need in your applications. And, OPAL's compiler gives you even greater machine efficiency, when required.

Functionally, OPAL is a flexible shell that offers a high degree of programming, featuring flow-of-control, Menu and Form screen definition, CALLs, DO statements, DATE and string functions, to name a few. OPAL is so complete, some users have used it for prototyping entire applications!

Circle no. 361 on reader service card.

Moreover, OPAL provides benefits through its synergy with DOS. OPAL understands how to work with and enhance your DOS environment.

If you feel encumbered and constrained by your current exec language, you NEED OPAL...

..only \$169

Texas residents add 6.125% sales tax Shipping \$4.00 MasterCard, VISA Accepted

To order or receive more information: (214) 490-0835

## The Software Factory

15301 Dallas Parkway Suite 750 LB 44 Dallas, TX 75248

IBM is a registered trademark of International Business Machines Corp OPAL is a trademark of The Software Factory, Inc.

## STRUCTURED PROGRAMMING (continued from page 113)

The first problem is getting a good name. But then, once good (or even merely tolerable) names have been arrived at, they must also be remembered somehow and (in a team situation) communicated. The Forth dictionary is not really a dictionary for humans: the names are not arranged alphabetically. Some versions of Forth provide words such as *LOCATE* or VIEW that work reasonably wellbut only if you can remember the name to begin with (does anyone have a LOCATE and VIEW that work with wildcards?) and if the documentation (comments and shadow screens) is understandable and up-todate—or, failing that, the source code is readable.

The difficulty of using new words fluently is the same as the difficulty of speaking or writing a foreign language. Having to look up every word in a dictionary is insufferably slow. In a single-programmer shop, the programmer gradually learns his or her own language and becomes fluent in the tools he or she has created. But what is to be done in a multiperson shop, with each programmer creating several names a day? Are there regular meetings wherein the programmers present their words to each other? Do they pass around a list of their creations for others to learn and use? Do they maintain an on-line encyclopedia? When a new programmer joins the group, how is that person trained in the local language? How long does it take a programmer new to the group to become fluent in the special words that are in use?

I am in the lone-programmer category, but I would be interested in hearing how multiperson shops handle the problem of names and the problem of promulgating the general-purpose tools the programmers create. If you have found a working solution to this problem, do share it.

I also would be interested in finding out how you lone wolves keep track of your own tools. Do you sort your tools into files, each file being a toolkit for a particular purpose? Do you use precompiled overlays as toolkits? Do you keep all your words and their use in your head, or do you maintain some kind of written refer-

ence book—a dictionary, or thesaurus, or encyclopedia? Let us in on your secrets.

#### Fragility as Strength

Once there was a contest to define Forth in 25 words or less. My definition was "Forth is like the Tao: it is a Way, and is realized when followed. Its fragility is its strength, its simplicity is its direction." I want to talk about the seeming oxymoron in this definition: Forth's fragility being its strength.

Forth has no training wheels. If you tip over, you fall: the stack explodes, the system crashes, whatever. The design decision in creating Forth was to remove safeguards to enhance performance. For programmers accustomed to bulletproof compilers, this approach seems foolhardy. Why not have as much protection as possible?

Protection of course imposes performance penalties, but perhaps even more important is the degradation of the feedback. In high-performance machines, the flip side of responsiveness is sensitivity. The more the machine gives control to the operator, the more responsibility the operator must accept. The advantage of the operator taking control is that the operator becomes more directly connected to what is happening. This connection amplifies awareness and allows the mind and the tool to merge, providing the immediacy of feedback that more closely connects thought and action. The intimacy and control of such a connection is almost addictive, which is why people who have learned to work with such tools are so reluctant to abandon them. Racing-car drivers don't enjoy spending the day behind the wheel of a station wagon.

Robert Berkey first pointed out to me how the Forth stack, leaving the arguments nakedly exposed, also lets the programmer see what is going on. Errors surface immediately—that's the fragility—and, being discovered, are then corrected—that's the strength. Merely because Forth is fragile for the programmer does not mean that the application programs are fragile. Indeed, the very degree to which errors will out during development makes the final product that much more robust.

Fragility often accompanies flexibility. The more options the machine or language offers, the more ways it can be used against itself (fragility), but the greater diversity of needs it can address and the more quickly it can be modified (strength). A mechanical example is the Gossamer Condor, a successful human-powered aircraft. A key design deci-

sion was not to attempt to make it an unbreakable machine but to make it as simple as possible, with everything visible and accessible. Let it break, as long as it is easy to fix. That simplicity also made it flexible in the sense that it was easy to modify, and in fact the Gossamer Condor's success was based upon a process of iterative development familiar to Forth pro-

```
1000 0000

No more data in this 512-byte block.

Data field consists of as many bytes as specified by the number in the low bit positions (and thus a maximum of 127 bytes). Though unimportant for decoding, it is worth noting that the data bytes contain no duplicates.

Data field consists of a single byte (the next byte after this flag), which is to be replicated as many times as the number in the low bit positions (and thus a maximum of 127 replications).
```

Table 4: Flag bit structure

```
(Work areas)
  CREATE OUTAREA 20000 ALLOT (will contain uncoded image)
  OUTAREA 20000 ERASE
                              ( size depends on application )
  CREATE INAREA 512 ALLOT
                              ( work area for input blocks )
(Pointers)
  VARIABLE INBYTE ( current byte in work area )
  VARIABLE OUTBYTE ( current byte in output area )
: INPOINT ( - adr ) INBYTE a INAREA +; ( next source byte )
: OUTPOINT ( - adr ) OUTBYTE a OUTAREA +; ( next target byte )
(Flagmanipulation)
(These use the encodingflags)
: NEXTFLAG
               ( - f)
                         INPOINT Ca;
                                           ( putsflag on the stack
: BLOCKEND?
              (f-f)
                         128 = ;
                                           ( end of input block )
: REPLICATE? (f - f)
                         128 AND :
                                           (replicate next byte)
: CHARCOUNT
             (f-n)
                         127 AND ;
                                           ( # of replications or )
                                           ( # of bytes to move )
(Replication)
: REPLICATE (f - )
                               (replicates based on count inflag)
   INBYTE INCR
                                (move past theflag)
   INPOINT Ca
                                 (char to replicate)
   OUTPOINT
                                ( destination address )
   ROT
                                 (bringflag to top)
   CHARCOUNT OVER + SWAP
                                 (indices = address range)
   DO DUP I C! OUTBYTE INCR LOOP
                                (replicating & counting)
   DROP
                                 (character replicated)
   INBYTE INCR ;
                                 (tonextflaglocation)
(Move)
: MOVECHUNK (f - )
                              ( moves # of chars specified inflag )
   CHARCOUNT DUP INBYTE INCR
   INPOINT OUTPOINT ROT CMOVE
                                 ( move characters )
   DUP INBYTE +! OUTBYTE +! :
                                 ( update pointers )
( Actual decoding of block )
: BLOCKWORK
                            ( decompress the run-length encoding )
   BEGIN NEXTFLAG DUP BLOCKEND? NOT OUTBYTE a 20000 ( AND
   WHILE DUP REPLICATE? IF REPLICATE ELSE MOVECHUNK THEN
   REPEAT DROP (flag);
```

Code Example 2: Decoding run-length encoded data

#### STRUCTURED PROGRAMMING (continued from page 115)

grammers. (The best account of the development of the Gossamer Condor and its sibling the Gossamer Albatross is the book Gossamer Odyssey by Morton Grosser [Boston: Houghton Mifflin, 1981).

In this spirit, tools for developers typically lack the safeguards that programmers provide in application programs: DROP, for instance, doesn't check stack depth before trying to drop. Such a check would slow it down too much. The programmer is responsible for making sure that the program will always have something on the stack when DROP is used.

On the other hand, some safeguards don't cost much. Paul Simon pointed out in a letter that the defining word FOR, which appeared in this column in July 1986, could include an error check with no speed penalty.

In its final version, FOR expects two numbers on the stack and will crash the system if it is executed with an empty stack. This behavior, perfectly acceptable when FOR was mine alone, becomes arguable when FOR is promulgated as a tool for general use. It is easy to provide some protection. The simplest approach is to include at the beginning of FOR's definition (right after CREATE) this phrase:

DEPTH 2 < ABORT" Need both array type and number of slots"

The speed of the words defined by FOR is unaffected by this additional check. On the whole, putting in this bit of protection seems reasonable. Moreover, as Forth is an open-architecture language, those who don't want to spend the memory space on the error message can remove the check. After all, they might reason, if FOR fails, it is during development. when the developer can immediately correct the condition. At run time (when the end-user is running the application program), it is not FOR but the words defined by FOR that are used, and they, of course, will work

Though I tested FOR for suitability as a tool for other programmers (as a tiny application program), I never recognized the problem of what happens when the stack is empty. My oversight occurred because I fell into the vulgar error of testing to show that the routine works instead of viewing as a failure any test that fails to find a bug.

Glenford J. Myers observes in The Art of Software Testing (New York: John Wiley & Sons, 1979) that the primary difference between successful and unsuccessful test efforts is that single, critical definition: a successful test is one that finds a bug; a test that finds no bug is a failure. And Gerald Weinberg's enjoyable book The Psychology of Computer Programming (New York: Van Nostrand-Reinhold Co., 1971) points out that a programmer trying to find errors in his or her own work is unlikely to be successful, which is why independent testing is so important.

#### **Run-Length Decoding**

I close the column with a brief discussion of run-length decoding. One way of compressing data is runlength encoding. There may be varieties of this technique, but the one I ran across was as follows.

The data are stored in 512-byte blocks, coded in variable-length data fields. Each data field has as the first byte a flag that determines the type of field and the length of the field. The flag's bit structure determines its meaning (see Table 4, page 115).

The words in Code Example 2, page 115, decode such encoded data. In this particular application I knew that the decoded data would not exceed a length of 20,000 bytes. Each coded block is read in turn into the 512-byte input work area and is then decoded into the next available area of the output work area.

The pointers INPOINT and OUT-POINT keep track of where you are in the two areas. The flag-manipulation words take care of all flag interpretation. REPLICATE replicates, and MOVE-CHUNK moves a chunk of data. BLOCKWORK decodes the block and is used within a loop that reads each of the 512-byte input blocks into the input work area in turn.

DDJ

Vote for your favorite feature/article. Circle Reader Service No. 7.

# Brand New From Peter Norton A PROGRAMMER'S EDITOR

that's lightning fast with the hot only

Direct from the man who gave you The Norton Utilities, Inside the IBM PC. and the Peter Norton Programmer's Guide. features programmers need "This is the programmer's editor that I wished I'd had when I wrote my

Easily customized, and saved Split-screen editing A wonderful condensed/outline display Great for assembler, Pascal and C

Peter Norton Computing, Inc., 2210 Wilshire Boulevard, Santa Monica, CA 90403, 213-453-2361. Visa, Mastercard and phone orders welcome.

> The Norton Editor™ is a trademark of Peter Norton Computing, Inc. ⊂ 1986 Peter Norton Computing Circle no. 243 on reader service card.



Norton Utilities. You can

program your way to

glory with The Norton

# DR. DOBB'S CATALOG

# just released!

# WRITING A UNIX-LIKE SHELL FOR MS-DOS

Allen Holub's new book includes an enhanced version of his popular Unix-like Shell. You'll learn how to write shells applicable to MS-DOS, as well as to most other programming environments!

## ALSO INSIDE:

M&T BOOKS

DR. DOBB'S COMPLETE C TOOLBOX

THE TOOLBOOK OF FORTH

Z80 TOOLBOOK

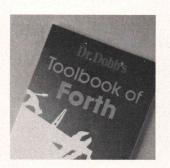
TURBO PASCAL TOOLS

DR. DOBB'S TOOLBOOK OF

68000 PROGRAMMING

TAMING MS-DOS

DR. DOBB'S BOUND VOLUMES









# NEW! DR. DOBB'S BOUND VOLUME 10

Bound Volume #10: The year of living dangerously. In 1985, iconoclastic DDJ beat Apple to the goal of adding more memory, a SCSI port, and a hard disk to the Macintosh. We dared to criticize the much-praised Turbo Pascal, challenged the Unix establishment with plans for a free Unix, and asked hard questions about privacy and control in the Information Age. Of course we also kept the technical level high, with the most exhaustive review ever of programmers' editors and of C compilers, and with powerful software tools in C, Modula-2, Forth. Pascal. assembly language, and Prolog.

Item #020D \$35.75

Bound Volume #1: 1976 The working notes of a technological revolution. Before there was an Apple, DDJ put a programming language on the first microcomputers, and became chronicler and instrument of the microcomputer revolution.

Item #013 \$30.75

**Bound Volume #2: 1977** Running light without overbyte. By year two the formula was clear: serious technical questions handled with a minimum of reverence; much source code; and a commitment to tight coding.

Item #014 \$30.75

**Bound Volume #3: 1978** The roots of Silicon Valley growth. The S-100 bus was hashed out in DDJ's pages. Steve Wozniak and others published in DDJ code that would help build an industry.

Item #015 \$30.75

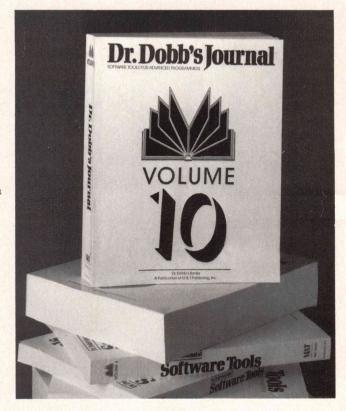
Bound Volume #4: 1979 In the midst of the gold rush. Three years before IBM moved in, the neighborhood was less civilized. DDJ published a gold mine of tips, tricks, and algorithms.

Item #016 \$30.75

**Bound Volume #5: 1980** C and CP/M. 1980 saw an all-CP/M issue, including Gary Kildall's history of CP/M, and Ron Cain's original Small-C compiler.

Item #017 \$30.75

Bound Volume #6: The First of Forth. This was the year DDJ launched its first Forth issue and Dr. Dobb's Clinic. Plus: PCNET, the Conference Tree, and 6809 Tiny BASIC. Item #018 \$30.75



**Bound Volume #7: 1982** Legitimacy. DDJ observed the IBM phenomenon, reviewed MS-DOS and CP/M-86, and looked forward to fifth-generation computers.

Item #019 \$35.75

**Bound Volume #8: 1983** Power tools. Professional software development on a PC was getting easier; DDJ helped, with Small-C, the RED editor, and an Ada subset.

Item #020 \$35.75

Bound Volume #9: 1984 Shaping things to come. In 1984 DDJ examined new programming environments: Prolog, expert systems, Modula-2, and a \$49.95 Pascal. Plus Allen Holub's GREP, Unix internals, and two encryption systems.

Item #020B \$35.75

## SAVE! ORDER THE COMPLETE 10 VOLUME SET!

Receive all ten Bound Volumes for only \$262! You save \$65!

Item #020E \$262



TO ORDER: RETURN THE FORM AT THE END OF THE CATALOG, OR CALL TOLL-FREE 1-800-528-6050 EXT 4001

AND REFER TO PRODUCT ITEM NUMBER, TITLE AND DISK FORMAT

# DR. DOBB'S C TOOLBOX

#### DR. DOBB'S TOOLBOOK OF C

SHENNE P

Over 700 pages of C material, including articles by such C experts as Kernighan and Ritchie, Cain and Hendrix, Skjellum and Holub! The level is sophisticated and pragmatic. The most valuable part of the Toolbook to many will be the hundreds of pages of useful C source code, including: Jim Hendrix's famous Small-C Compiler and New Library for Small C-Also available on disk!; NEW! Hendrix's Small Mac: An Assembler for Small C and Small Tools: Programs for Text Processing—Both also available on disk!; and all of Anthony Skjellum's C Programmer's Notebook columns distilled by Tony into one thought-provoking chapter.

From M&T Publishing and Brady Communications

Dr. Dobb's Toolbook of C Item #005 \$29.95

#### SMALL-C COMPILER

Jim Hendrix's Small C Compiler is the most popular piece of software published in Dr. Dobb's 11-year history. Like a homestudy course in compiler design, the Small-C Compiler and the Small-C Handbook provide everything you need but the computer for learning how compilers are constructed, and for learning C at its most fundamental level.

**Small-C Compiler** 

Item #007 \$19.95

#### THE SMALL-C HANDBOOK

Jim Hendrix's Small-C Handbook is the reference book on his Small-C Compiler. In addition to describing the operation of the compiler, the book contains complete source listings to the compiler and its library of arithmetic and logical routines. A perfect companion to the Hendrix Small-C Compiler available from DDJ on disk, the Handbook even tells you how to use the compiler to generate a new version of itself!

While both the Handbook and the Toolbook provide documentation for the Small-C Compiler, the Handbook contains a more detailed discussion and is available with addendum for the MS/PC-DOS version.

From M&T Publishing and Brady Communications

The Small-C Handbook Item #006 \$17.95

The Handbook with MS/PC-DOS Addendum

Item #006A \$22.95

#### C DISK FORMATS

When ordering, please indicate MS/PC DOS or CP/M. For CP/M disks, please specify one of the follwing formats: Apple, Osborne, Kaypro, Zenith Z-100 DS/DD, 8" SS/SD. Special order formats are available for an additional \$10 each.

#### SMALL-MAC: AN ASSEMBLER FOR SMALL-C

This assembler features simplicity, portability, adaptability, and educational value. The package includes: a simplified macro facility; C language expression operators; object file visibility; descriptive error messages; and an externally defined instruction table.

You get the macro assembler, linkage editor, load-and-go loader, library manager, CPU configuration utility, and a utility to dump relocatable files. Documentation is also included. For CP/M systems only. Please specify format.

Small-Mac Item #012A \$29.95

### SMALL-TOOLS: PROGRAMS FOR TEXT PROCESSING

This package of programs performs specific, modular operations on text files, including: editing; formatting; sorting; merging; listing; printing; searching; changing; transliterating; copying; concatenating; encrypting and decrypting; replacing spaces with tabs and tabs with spaces; counting characters, words, or lines; and selecting printer fonts.

Small-Tools is supplied in source code form only. With the Small-C Compiler you can select and adapt these tools to meet your own needs. Documentation is included.

**Small-Tools** Item #010A \$29.95

# SPECIAL PACKAGES 20% OFF

#### CP/M C PACKAGE

Receive this special package and save \$20! You'll get: Dr. Dobb's Toolbook for C; The Small-C Handbook; The Small-C Compiler on disk; The Small-Mac assembler on disk, with documentation; and The Small-Tools textprocessing programs on disk, with documentation all for only \$99.95!

Please specify format.

CP/M C Package

Item #005A \$99.95

#### MS/PC-DOS C PACKAGE

Save \$20 when you order this special package. You'll receive: Dr. Dobb's Toolbook of C; The Small-C Handbook with the MS/PC DOS Addendum; The Small-C Compiler on disk; and The Small Tools text-processing programs on disk, with manual all for only \$82.95. \$82.95

MS/PC-DOS C Package

Item #005B

TO ORDER: RETURN THE FORM AT THE END OF THE CATALOG, OR CALL TOLL-FREE 1-800-528-6050 EXT 4001 AND REFER TO PRODUCT ITEM NUMBER, TITLE AND DISK FORMAT



N SU MU MU MU MU

# TURBO ADVANTAGE: SOURCE CODE LIBRARIES FOR TURBO PASCAL

Here's the advantage you need to make your programming easier and more efficient! This library of 220 routines, complete with source code, sample programs and documentation, will save you hours developing and optimizing your programs. A few of the helpful **Turbo Advantage** files include:

- \* arithmetic operations \* bit manipulations \* check routines
- \* data compression \* differentiation and integration \* Fourier analysis and synthesis \* file management and hash methods
- \* Input/Output routines \* matrix processing \* menu functions
- \* MS-DOS support \* linear and nonlinear optimization
  \* screen/printer redefinition \* sorting routines \* spline
  integration, differentiation, interpolation \* statistical
  procedures \* string processing \* type conversion
  All procedures and functions are supplied in source code form

so you can adapt them to meet your needs. Each file contains a list of relevant routines, all explained with example programs to help you understand the callings, parameters and data types in detail. Most of the programs can be immediately compiled. The detailed manual includes a short characterization of the routine, a description of the way the routine works with an explanation of the methods used, the calling sequence—including a description of the parameters and the function result, notes with advice and special explanations, and a simple example illustrating the files and variables needed.

Turbo Advantage is for MS/PC DOS systems. Item #070 \$49.95

### TURBO COMPLEX ADVANTAGE: COMPLEX NUMBER ROUTINES FOR TURBO PASCAL

Working with complex numbers, vectors and matrices is easy with **Turbo Advantage Complex!** This library of over 80 procedures and routines will help you program complicated functions more easily. You'll find routines to help you:

- \* use digital filters, solve boundary-value problems and process very large arrays;
- \* carry out vector and matrix calculations with complex integers and variables;
- \* execute time-saving simultaneous Fourier transforms and calculations of convolution and correlation functions.

The Turbo Complex package includes source code and documentation. For MS-DOS systems. Some of the Turbo Complex routines are most effectively used with routines contained in the Turbo Advantage package.

Item #071 \$89.95

### TURBO DISPLAY ADVANTAGE: FORM GENERATOR FOR TURBO PASCAL

Now, even if you have little programming knowledge, you can design and process forms to fit your needs! The Turbo Display form generator includes a text editor, 30 time-saving Turbo Pascal procedures and functions, and a concise, informative handbook to help you take full advantage of all Turbo Display capabilities.

Turbo Display includes source code and documentation. For MS-DOS systems. Some of the Turbo Advantage routines are necessary to compile Turbo Display.

Item #072 \$69.95



## THE TURBO PASCAL TOOLBOOK

The Turbo Pascal Toolbook contains an extensive library of routines and sample programs to help make your programming easier and more powerful. Let Turbo Pascal experts show you how to easily integrate the routines into your own programs. Innovative sample programs demonstrate exactly how the routines are implemented.

THE RESERVE

- Mathematical Expression Parsers offers two routines that convert mathematical expressions into RPN-tokens.
- The Spiderweb Database Structure presents a new database routine that combines the best features of the B-tree, B+ and B++ trees.
- When Turbo Isn't Enough presents an extensive library of low-level routines that show you how to tap into the hardware and the operating system.
- Artificial Intelligence Techniques includes three programs that demonstrate ways to implement a user-friendly system.
- Window Management will help you create, sort and overlay windows.
- A Smart Regression Model Finder automatically searches for the best regression model to analyze a given set of data.

The routine libraries and sample programs in **The Turbo Pascal Toolbook** are also included on disk for MS-DOS
systems. All source code is included.

The Turbo Pascal Toolbook Item #080 \$25.95 The Turbo Pascal Toolbook with disk Item #081 \$45.95

## STAT TOOLBOX FOR TURBO PASCAL

The STAT TOOLBOX is two statistical packages in one! Whether you're a programmer looking for tools to build your own applications, or a user who wants a complete, fully functioning package, the STAT TOOLBOX will bring convenience, power and versatility to your statistics programs! The STAT TOOLBOX is written in Turbo Pascal and includes full source code. It contains two complete packages:

#### A reference disk and manual

Flexible, time-saving building blocks allow you to customize statistical applications to fit your needs. You'll find:

- statistical distribution functions;
- random number generation;
- basic descriptive statistics
- parametric and non-parametric statistical testing.
- bivariate linear regression, multiple and polynomial regression
- · automatic best curve selection

#### A demonstration disk and manual

The demonstration package includes fully functioning statistical programs, and two data management programs. In addition to these practical tools, files containing sample data will help users learn to enter, edit, maintain and manipulate data. High resolution graphics capabilites include drawing histograms and regression lines, and plotting residuals for regression analysis. (Borland's Turbo Graphic's Toolbox is required) For IBM PC's and compatibles. Turbo Pascal version 2.0 or later, and PC-DOS 2.0 or later are required.

Stat Toolbox Item #050 \$69.95



# THE DR. DOBB'S TOOLBOOK SHELF

## Z80 TOOLBOOK

David E. Cortesi, longtime Dr. Dobb's columnist brings you:

• a method of designing programs and coding them in assembly language. Cortesi walks through the initial specifications, designing an algorithm and writing the code. He demonstrates the construction of several useful programs.

- a complete, integrated toolkit of subroutines for arithmetic, for string-handling, and for total control of the CP/M file system. They bring the ease and power of a compiler's runtime library to your assembly language work, without a compiler's size and sluggish code.
- · Every line of the toolkit's source code is there to read.

#### ORDER THE Z80 SOFTWARE ON DISK!

All the software in Dr. Dobb's Z80 Toolbook—the programs plus the entire toolkit, both as source code and object modules for both CP/M 2.2 and CP/M Plus—is yours on disk! Most of the programs are included in the book, however, the disk is necessary for complete listings. A Z80 microprocessor and a Digital Research International RMAC assembler or equivalent are required.

Dr. Dobb's Z80 Toolbook Item #022 \$25 Dr. Dobb's Z80 Toolbook w/disk Item #022A \$40 Please specify one of the following disk formats: 8" SS/SD, Apple, Osborne, or Kaypro

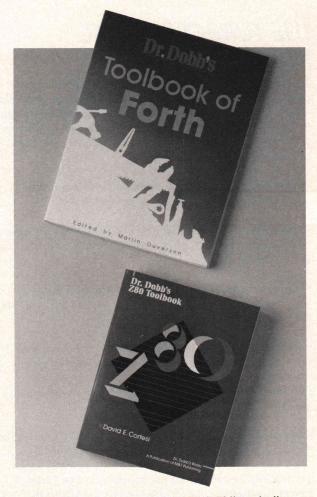
## DR. DOBB'S TOOLBOOK OF FORTH

This comprehensive collection of useful Forth programs and tutorials contains DDJ's best Forth articles, expanded and revised along with new material. In addition, you'll glean important insights about the potential of this increasingly popular language from the many in-depth discussions of advanced Forth topics. You'll find sections on:

Mathematics in Forth, including "Series Expansion in Forth," "Forth Floating-Point Package," and "Signed Integer Division"

Modifications/Extensions, including "A Proposal for Strings in Forth," "Non-Deterministic Control Words," "Some Forth Coding Standards," and "Towards a More Writable Forth Syntax"

Forth Programs, including "GO in Forth," "Elements of a Forth Data-Base Design," "The Forth Sort," "SEND & RECV," "Interface for a Mouse," "Relocating Loader in Forth," "Forth Decompiler," "Screen-Oriented Editor ReVisited," "Evolution of a Video Editor," "H-19 Screen Editor," and "The Conference Tree."



Forth—the language, including "The Forth Philosophy," "Teaching Forth as a First Language," and "Forth-83 and Vocabularies"

Implementing Forth, including "Forth and the Motorola 68000," "A 68000 Forth Assembler," "A Forth Assembler for the 6502," and "Z8000 Forth." You'll also find Appendices that will help you convert fig-Forth to Forth-83, and tell you how to stay up-to-date on the latest developments and refinements of this popular language.

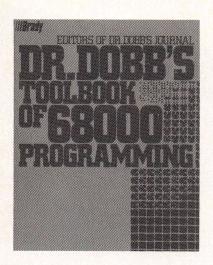
The screens in the book are also available on disk as ASCII files. Receive Dr. Dobb's Toolbook of Forth, along with the software on disk, together for only \$39.95.

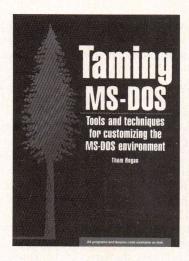
Dr. Dobb's Toolbook of Forth Item #030 \$22.95 Dr. Dobb's Toolbook of Forth w/Disk

Item #031 \$39.95

Please specify MS/PC-DOS, Apple II, Macintosh, or CP/M. For CP/M disks, specify Osborne or 8" SS/SD.

# THE DR. DOBB'S TOOLBOOK SHELF





## DR. DOBB'S TOOLBOOK OF 68000 PROGRAMMING

In this complete collection of practical programming tips and techniques for the 68000 family, you'll find the best articles on 68000 programming ever published in Dr. Dobb's, along with new material from 68000 experts. You'll learn about the most important features of the 68000 microprocessor from a full description and concise discussion of its history and design. And, useful applications and examples will show you why computers using the 68000 family are easy to design, produce and upgrade. Contents include:

#### an Introduction to the 68000 Family

• 68000 Instruction Set

#### **Development Tools**

- Bringing Up the 68000: A First Step
- A 68000 Cross-Assembler

#### **Useful 68000 Routines and Techniques**

- A Simple Multitasking Kernel for Real-Time Applications
- · The Worm Memory Test
- A Mandelbrot Program for the Macintosh

### All programs are available on disk!

In addition, an executable version of the 68000 Cross-Assembler can be purchased along with the source code and documentation for \$25. (Requires one or more disk drives and either CP/M-80, CP/M 2.2 with 64 bytes or MS-DOS with 128

Dr. Dobb's 68000 Programming Toolbook Item #040 \$29.95 68000 Toolbook with Disk Item #041 \$49.95

Please specify one of the following disk formats: CP/M 8", Osborne, Macintosh, Amiga, Atari 520st, MS-DOS

68000 Cross-Assembler Item #042 \$25

## TAMING MS-DOS BY THOM HOGAN

Learn how to make DOS work for YOU! Taming MS-DOS will take you beyond the basics, picking up where your DOS manual leaves off. You'll find batch files and DOS enhancements that extend the power of DOS so you can work more accurately and efficiently. And, you'll learn how to customize DOS to fit YOUR needs, saving you time and frustration every time you use it.

Taming MS-DOS will show you how to create a memory-resident clock, rename subdirectories and change file attributes. You'll learn more about what's on your disk and how to protect it easily, how to create configurable AUTOEXEC.BAT files and understand redirection and piping. You'll find:

- how to customize CONFIG.SYS and use ANSI.SYS to change the appearance of DOS
- extensive batch file coverage
- · example routines using redirection, including redirection operators, filters, and pipes
- · DOS enhancement programs that are ready to use, with executable commands.
- Full source code, allowing you to alter programs and create a custom system.

The book includes assembly language programs that allow you to manipulate DOS at its lowest form. An assembler is not needed to enter the programs; Taming MS-DOS presents an alternative method for anyone with BASIC on their machine. The programs, including batch files and DOS enhancements are also available on disk along with source code.

**Taming MS-DOS** 

Item #060

\$19.95

Taming MS-DOS with disk

Item #061

\$34.95



# WRITING A UNIX-LIKE SHELL FOR MS-DOS

This book will show you how to write shells applicable to MS-DOS as well as to most other programming environments. The book and disk include an in-depth description and enhanced version of Holub's popular Unix-like Shell, along with complete C source code. You'll find:

THE REPORT

• how to do interpretive control flow in any C program;

• a thorough discussion of low-level DOS interfacing;

• significant examples of C programming at the system level. The Shell's NEW supported features include:

NEW! READ Lets you use input from the keyboard from within the Shell script.

NEW! A new Shell variable expands the contents of a file so a program can produce text that is used inside of the Shell script.

Editing Command-line editing with the cursors is supported. The line is visible as you edit it.

Aliases Can be used to change the names of commands or as very fast, memory-resident, batch files. Nested aliases are supported.

History You can execute previous commands. The command can be edited before being executed. Imbedded history requests (Bar; !!>foo) are supported.

Redirection and Pipes <>>> & >> & |

Pipe temporary files can be put on a RAM disk.

Unix-like Command Syntax/ can be used to separate directory names (\ can now be used as well). A 2048-byte command line is supported. Command-line wild card expansion. Multiple commands on a line.

#### DOS-compatible prompt support

\$d \$t \$1 \$h \$n \$q \$\$ \$%

C-Shell Based Shell Scripts (batch files) Shell Variables are macros that can be used on the command line. Arithmetic manipulation of shell variables using the @ command are supported. The following C operators are also supported: () + -\*/% <=>=<>!===! & & #=

A batch file can call another batch file like a subroutine. Control is passed to the second file an then back to the first when the second is finished. Batch files can return values to the calling file using the exit and \$status mechanisms.

A powerful, interpretive, programming language, based on the UNIX C Shell, is now supported, including:

if/then/else foreach break
while switch/case continue
All commands can be nested.

The shell runs on IBM PC's and compatibles.

Writing a Unix-like Shell for MS-DOS book & disk Item #163

\$39.95



## /UTIL

/Util is a collection of UNIX-like utility programs for MS-DOS. This package includes updates of the highly acclaimed Dr. Dobb's articles; Grep: a UNIX-like Generalized Regular Expression Processor, and LS and Getargs from DDJ's C Chest.

Source code is included and all programs (and most of the utility subroutines) are fully documented in a UNIX-style manual. You'll find executable versions of:

cat	echo	mv	rm
ср	grep	p	rmdir
date	ls	pause	sub
du	mkdir	printenv	chmod

/Util Item #161 \$29.95

# ORDER FORM

#### **ORDER NOW!**

NAME (Please use street addr ADDRESS	ress, not P.O. Box)		-		7.0	00000		
CITY		STATE ZIP		ZIP	TO ORDER: CALL TOLL FREE 800-533-437			
DAY PHONE						Fri, 8-5 Pacific Time) alif: 800-356-2002		
	For disk orders, p availability for each	olease indicate format. R product. Special forma	efer to ad for stand its available for addi	ard format tional \$10 each.		our order and payment to: alveston Dr., Redwood City, CA 940		
	MS/DOS cintosh		Zenith Z-100 Osborne	DS/DD  — Amiga — Atari 520st	MasterCard	CATES		
QUANTITY		DESCRIPTION	Apple		UNIT PRICE	TOTAL PRICE		
				100				
		on merchandise total		<b></b>	SUB-TOTAL SALES TAX			
					SHIPPING TOTAL ORDER			
П	VISA	NAME ON CARD						
	ASTERCARD	ACCOUNT NO.						
	RICAN EXPRESS	EXPIRATION DATE						
□ СНЕСК	(make checks payable to M&T Publishing)	SIGNATURE				-		

In U.S. For Bound Volumes, add \$3.25 per book. Add \$9.25 for Special C Packages. For other books and disks, add \$2.25 per item. Outside U.S. For Bound Volumes, add \$6.25 per book surface mail. Add \$18 surface mail for Special C Packages. For other books and disks, add \$5.25 per item surface mail. Foreign airmail rates available on request.

For Faster Service and reduced shipping costs, Europeans may order direct from: Markt & Technik, Buchverlag, Hans-Pinsel-Strasse 2, 8013 Haar bei München. Call Germany 89-4613-221 for prices in Deutch Marks.

# DR. DOBB'S CATALOG ORDER FORM

Please fold along fold-line and staple or tape closed.



No Postage Necessary If Mailed In The United States

## **BUSINESS REPLY MAIL**

First Class Permit No. 790 Redwood City, CA

Postage Will Be Paid By Addressee

DR.DOBB'S CATALOG

501 GALVESTON DRIVE REDWOOD CITY, CA 94063

Please fold along fold-line and staple or tape closed.



### is at hand

HELP/Control™ - an on-line help subsystem for the IBM-PC.

Increases the value of your software. Save development time and money

HELP/Runtime. A few simple subroutine calls add context sensitive on-line help to your application. HELP/Runtime includes tested interfaces for Microsoft C, Lattice C, Turbo Pascal, IBM BASIC (Interpreter and Compiler), Microsoft FORTRAN, IBM COBOL and assembler. It is distributed with demonstration programs in each language.

HELP/Popup. Add a powerful help system to existing applications, even in dBase or 123, without reprogramming, even without a programmer. It may be memory resident, or, installed with an application, it terminates when the application exits, releasing its memory.

HELP/Generation. Use your favorite editor and our concise screen definition language to build your help files. Compile them into a help system usable by either HELP/Runtime or HELP/Popup. The package includes sources for sample help files illustrating such features as full-sized or windowed screens.

HELP/Convenience. The screens include highlighted captions. The user selects a caption with the cursor control keys and advances to a new screen, just as with 123.

HELP/Documentation. A detailed manual, both on-line and printed, for the documentation writer and programmer includes instructions which may be incorporated into the user manual.

HELP/Environment. PC-DOS 2.0 or greater is required. HELP/Runtime requires approximately 9K for code and buffers for full size help screens

HELP/Pricing. The complete package (software, both manuals, and demo programs) costs \$125.00 and includes a royalty-free license to add HELP/Runtime to your applications and to make 25 copies of HELP/Popup. A demonstration diskette, including the on-line manual, costs \$15.00. A free update to Release 1.1 is available to registered owners. To order, or for more information (including dealer, multiple-copy and site-license pricing) call MDS at 207/772-5436. We accept MasterCard and VISA.



MDS, INC., P.O. BOX 1237, PORTLAND, MAINE 04104

Circle no. 285 on reader service card.

# Tom Rettig's Library

Prewritten Solutions to Programming Problems

### Clipper Edition

#### Advanced Extended Library

- · Convenient .LIB library file to link
- Separate object files so only the code actually used is added to your program
- Optimized for speed and code size
- Requires Clipper, Winter '85 or later

#### dBASE III PLUS Edition

#### Advanced Programmer's Library

- Interface to C/assembler is fully compatible with Clipper's Extend system
- Functions are CALL procedures
- Requires dBASE III PLUS, any version, and 128K of additional memory

#### Each edition comes complete with...

- Entire source code, nothing withheld. Cleanly written, liberally commented, easy to modify, helpful for learning.
- Support by phone and electronic mail
- No royalties or copy protection
- Documentation inserts for dBASE or Clipper manual. Accurate, complete, easy to use, page-per-command.
- · Handy plastic reference card
- Full money-back guarantee

Over 175 functions; 55% written in C, 35% in Assembler, 10% in dBASE

\$99.95 per edition at dBASE/Clipper dealers or direct

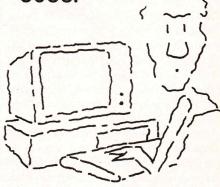


9300 Wilshire Boulevard, Suite 470 Beverly Hills, CA 90212-3237, U.S.A. (213) 272-3784 Telex: 4996426 RETTIG Source: BCR480 
CompuServe: 75066,352

dBASE and dBASE III PLUS are Ashton-Tate trademarks, Clipper is a Nantucket trademark Call or write for free product information

# Get a Grip on Assembly Language.

The award winning **Visible Computer:** 8088.



The Visible Computer is a book and software combination for mastering the elusive skills of assembly language. PC Tech Journal took one look and made it their September '85 "Program of the Month."

It's an animated simulation of the PC's microprocessor that lets you see with your own eyes how assembly language works. You'll be using it as a debugging tool for years to come.

It's a tutorial. A lot of people think the 350 page manual is the best book on assembly language ever written.

It's 45 demonstration programs you'll execute with the simulator, from simple register loads to advanced programs that manipulate interrupts and perform file I/O. And what you'll learn applies to all 86 family processors, including the 80186 and 80286.

The Visible Computer for IBM PC/XT/AT and true compatibles. If your dealer doesn't have it, order direct: Software Masters, 2714 Finfeather, Bryan, TX, 77801. (409) 822-9490. Please include \$3.00 shipping. Bank cards accepted.



TVC takes you inside the processor as it executes programs.

**Software Masters** 

#### A New Project Is Born

I'd like to design a versatile, easyto-use interpreted language, using occasional essays in this space to stimulate my own creative juices and get feedback from you. My approach to this project will be experimental, and the entire interpreter will be written in 680xx assembly language. Why? Because I love 680xx assembly language, and I like to noodle around looking for really efficient ways to do stuff. As my interpreted language comes together, I want to know what you think of it. If the ideas expressed here get your juices flowing, send me a letter. If you send me interesting enough letters, I'll include them here. I'd like this to be both an educational project for interpreter designers and a general discussion of data handling in assembly language.

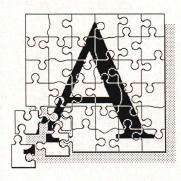
## Why an Interpreted Language?

The nicest thing about an interpreted language is that it can be very interactive, and if it's extensible and fast enough, it can be a real joy to work with. Forth is an example of the kind of language I'm talking about. I like Forth a lot. But the problem with Forth is that it's too weird—I find I have to think backward to use it effectively, and I'd like to be able to think in my most efficient way—forward. Thus, because I've never seen a true interpreted language that satisfies me, I want to design my own, with your feedback to help guide me.

#### by Nick Turner

#### **Juggling Numbers**

In this first essay I want to talk about math; specifically, numeric formats. This is intended as both an introduction to numeric representation (for those who may not have a lot of low-level practice) and as a source of inspiration for experienced assembly-language programmers. I'll start



things off with a summary of some of the various numeric formats that have been used on computer systems. This will be a general description; more detailed stuff will come later on. I hope to make most of these formats available in the final interpreter.

#### **Simple Integers**

The simplest approach to computer math is to use integers. Though integer (INT) math may initially seem rather limited, a surprising amount of complex calculation can be done with integers alone. On typical computer systems, there are usually three kinds of integers: bytes, words (two bytes), and long words (two words). Sometimes you might need extra-precision integers of eight or more bytes. I propose at least two kinds of integers for my interpreter: word size (INT) and long word size (LINT). Both would be signed values, with negative numbers expressed in two's-complement form. Will I need double-long, 8-byte integers (DINT)?

Simple math with integers is straightforward. The biggest advantage of INT math is speed. Overflow and underflow are typically the most important error conditions. The biggest practical disadvantage of integer math is the inability to represent fractional values directly. Fractions can be represented by multiplying all the numbers in the system by some constant, but it requires extra time and programming. Besides, if the constant multiplier is a power of 2, you've just invented the next category: fixed-point numbers.

#### **Fixed-Point Numbers**

A typical fixed-point (FIX) representation allocates a number of bits for the integer portion of a value and an equal number of bits for the fractional portion. For example, you might use a 4-byte long word in which the highorder word is the integer and the loworder word is the fraction. Some systems use larger FIX formats with a whole long word for each portion, and a few systems have unequal distributions of bits. In such cases, it's usually the fractional portion that has fewer bits. I propose one FIX format for my interpreter (mostly for speed in calculations involving fractions). My FIX could be two long words-one for the integer and one for the fraction. The high bit of the integer portion would be reserved for the sign, and the rest would be an unsigned value. (This simplifies output of ASCII translations of the number.)

FIX has the advantage of being able to deal with fractions, but it still has the problem of limited precision, especially for small numbers. From here there are two directions in which to go. Which path a system takes depends on what the numbers will be used for. If the ability to represent really huge or miniscule values is more important than vastly precise representations, then floating point is probably best. On the other hand, if incredibly high precision is necessary, you might choose what I call extended representation.

#### **Floating-Point Numbers**

By far the most frequent choice in typical systems is a floating-point representation (FLOAT), in which the value is divided into two subvalues: the exponent and the mantissa. The exponent represents the logarithm in base 2 of a number by which the mantissa is to be multiplied to create the actual value stored. For example, if the exponent is 4 and the mantissa

is 3, then the value might be 3 times 2 to the fourth power, or 3 times 16, or 48. In actual practice, the mantissa is almost always treated as a fraction. In the above case, the exponent would be 6 and the mantissa would be 0.11 (binary), which is 3 (or 11 binary) shifted left twice. Note that the exponent really represents nothing more than the number of times the mantissa must be shifted to create the actual value. If the exponent is negative, you shift the mantissa to the right. If it's positive, you shift it left. The mantissa usually also has a sign bit, which governs the sign of the entire value.

Now here's the tricky part about floating point: most FLOAT representations nowadays have something called a "hidden 1 bit." This means that the high-order bit of the mantissa, which is always a 1 bit in a properly normalized FLOAT value, is "overlaid" by the sign bit of the mantissa or is omitted altogether. The cost of this 1-bit saving is that the missing bit must be recreated every time a calculation is done. For systems with a hardware assist, such as the MC68881 floatingpoint math chip, this is trivial. Another tricky point is that the exponent is usually represented as an "augmented" value-this means you must first subtract a certain number from it in order to get the actual exponent. The augment number is chosen such that an exponent of zero is represented as a bit field with only the high bit set. The result is that the exponent can be treated as a simple unsigned value, simplifying many calculations.

For my interpreter, I propose the FLOAT formats used by the MC68881 chip—specifically, the single, double, and extended representations, which I will call FLOAT1, FLOAT2, and FLOATX for my language. The reason is simple: I'd like to use the 68881 chip eventually.

#### A Weird Extended Hybrid

The last approach to numeric representation, and one that I've not seen used very much, is sort of a weird hybrid between floating point and fixed point. I call it extended representation (EXT), and it's the only numeric format in my proposed system that uses variable-length fields. The basic concept is simple: a number is represented in full precision as a large field of 2-byte words, with a giv-



# Monster Mac-the high performance upgrade from Levco

**Speed Boost** MonsterMac boosts the speed of your Macintosh by 25-30%!

Extra Memory With up to 2.0 megabytes of contiguous RAM, you can use applications like MacWrite,™ Helix,™ Excel,™ Jazz,™ and PageMaker™ simultaneously, or create a 1.8-meg RAM Disk for a super-productive Macintosh

Proven Compatibility Fully compatible with original 64K ROMs and the current 128K Apple ROMs. Operates standard Macintosh applications without the handicap of configuration software.

#### **Apple-Standard SCSI Support**

MonsterMac's on-board SCSI Controller provides a speedy interface to Macintosh Plus-compatible hard disk drives. Levco's 20-meg *OverDrive* is also available for those who want a reliable, internal hard drive.

Easy Clip-on Installation! With Levco's secure, patented Chip-Clip™ you can install the MonsterMac quickly and easily on a 128K or 512K motherboard. Requires no special skills or modifications to your Macintosh.

Cooler Running Unique, motorless MacBreeze fan is included to keep your MonsterMac quietly cooler.

Reliability You can have confidence in a MonsterMac. With over 2000 satisifed owners, the original 2 meg upgrade is a proven performer.



#### Levco

6160 Lusk Blvd. Suite C-203 San Diego, CA 92121 (619) 457-2011

Macintosh\* is a trademark licensed to Apple Computer. MacWrite\*, Helix\*, Excel\*, Jazz\*, and PageMaker\* are trademarks of Apple Computer, Odesta Corp., Microsoft, Lotus Development, and Aldus respectively. MonsterMac, OverDrive and MacBreeze are trademarks of Levco.

Circle no. 346 on reader service card.

# C, BASIC, Pascal, dBASE, Modula-2

Source Print makes your job easier by clarifying your source code!

For the new low price of \$97, you get all these valuable time saving features: The Index (cross-reference) lists variables, functions, procedures, and fields. Structure Outlining draws lines around nested structures for you. Automatic Indentation keeps listings and source code uniform.

"here is possibly the ultimate source code printing

-Data Based A

"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to he programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."

> \_PC Magazine Sept. 16, 1986

The Table of Contents lists functions and procedures. Key words can be printed

in boldface. Functions and procedures can be extracted to build a new source file, or when printing. Multistatement BASIC lines can be split for readability.

The easy-to-use menu requires no learning period. Scroll thru directories. Search for files containing a given string great for finding that "lost" procedure.

# Tree Diagrammer shows you the forest as well as the trees!

Our new TREE DIAGRAMMER, for only \$77, automatically prints an organization chart of your program showing the hierarchy of calls to

functions, procedures, and subroutines. It's easy to see what's called from what.

Recursive calls are indicated.

Why not order these indispensable tools today? We ship immediately, and there's no risk with our 60-day moneyback guarantee.

Order by phone or mail:

800-257-5773, 800-257-5774(CA).

or see your local dealer. MC, Visa, AmEx, COD. Add \$5/order shipping. In CA add 6% tax.

ana emito closetxt proccode getcode putcode endout

main

# Laboratories Inc.

Mainframe-quality Software for the PC 3339 Vincent Rd., Pleasant Hill, CA 94523 415-930-8966

SOURCE PRINT and TREE DIAGRAMMER handle up to 50 source files and 60,000 program lines. For IBM PC and all compatibles, 256K.

#### THE RIGHT TO ASSEMBLE (continued from page 127)

en number of words representing the integer portion (except the highest order bit, which is the sign bit) and the remainder representing the fractional part. Of course, there must also be a field somewhere that contains some clue as to where the radix point is (the radix point separates the integer from the fraction). It's also important to have a value that says how long the whole thing is.

The EXT format has certain advantages for a limited set of problems. For instance, I've always wanted to be able to compute various irrational values to an arbitrarily high precision. My EXT format can do this, but problems arise. For example, as soon as you attempt to calculate a transcendental function, you run into precision vs. time trade-offs: if you use the traditional polynomial approximation method, your polynomial factors will limit the precision of the result, which must then be chopped accordingly. On the other hand, if you use the full Taylor (or similar) series to compute the transcendental result, you may end up spending an inordinate amount of time to get the desired accuracy. I'm very interested in feedback on this issue; I have by no means reached a satisfying resolution.

#### Do You Want More?

If there's a good response to this essay, I'll continue the story. Future topics might include a detailed expansion on each of the numeric formats described here, with listings of working math routines and a discussion of the "housekeeping" information surrounding the number formats—how does the system know what kind of number it's dealing with and how does it keep track of all the variables? I'd also like to discuss the actual syntax and interface of the language-but first I'd like to see your blue-sky suggestions. What would your ideal interpreted language look like? Write to me care of DDJ.

DDJ

Vote for your favorite feature/article. Circle Reader Service No. 8.

New Reduces

# Windows, Data Entry, Help Management, Menus, Text Editing, plus ...

# SOURCE CODE

# Vitamin C

It's good for your system!

#### The Vitamin C Difference

With Vitamin C, your applications come alive with windows that explode into view! Data entry windows and menus become a snap, and context sensitive pop-up help messages are nearly automatic.

With VCScreen, you'll save time by interactively painting windows and forms so what you see is what you get! Then, one button generates C source code ready to plug into your program and link with Vitamin C.

Easy enough for the beginner. Versatile enough for the professional. Vitamin C's openended design is full of "hooks" so you can intercept and "plug-in" special handlers to customize or add features to most routines.

Of course, Vitamin C includes all source code FREE, with no hidden charges. It always has. That means you'll have everything you need to adapt to special needs without spending hundreds of dollars more.

#### Windows

Create as many windows as you like with one easy function. Vitamin C automatically takes care of complicated tasks like saving and restoring the area under a window.

Options include titles, borders, colors, popup, pull-down, zoom-in, 4-way scrolling, scroll bars, sizes up to 32k, text file display & editing, cursor display, and more.

Unique built-in feature lets users move and resize windows during run-time via a definable

Access the current window by default or a specific window any time, even if it's hidden or invisible. Save and load windows on disk for more versatility!

#### Data Entry

Flexible dBase-like data entry and display routines feature protected, invisible, required. and scrolling fields. Picture clause formatting, full color/attribute control, selection sets, single field and full screen input, and unlimited data validation via standard and user definable routines. That means you aren't locked into one way of doing things.

Vitamin C even provides true right-to-left input of numeric fields with dynamic display of separators & currency symbols

#### High Level Functions

Use our intergrated help management, multi-level menus, and text file routines, or build your own handlers using Vitamin C's basic windowing and data entry routines.

Standard help handler provides context sensitive pop-up help messages any time the program awaits key strokes. The help text file is stored on disk and indexed for quick access. So easy to use that a single function initializes & services requests by opening a window, locating, formatting, displaying, and paging through the message.

Multi-level "MacIntosh" & "Lotus" style menus make user interfaces and front ends a snap. Menus can call other menus, functions, even data entry screens, quickly and easily.

Text editor windows can be opened for pop-up note pads, memo fields, or general purpose editing. Features include insert, delete, word wrap, and paragraph formatting.

### **VCScreen**

#### Screen Painter/Code Generator

Just as Vitamin C's reusable functions speed your programming, VCSreen makes it even faster and easier by automatically generating C source code for your data entry screens!

With VCScreen's interactive screen editor, you actually draw your forms. You can define input, output and constant fields, headings, boxes, lines and even a window for the form to run in

What you see is what you get. If you don't like the position of an object, just "pick it up" with the cursor and move it! Changing colors, attributes, copying, and deleting is just as

VCScreen generates readable C source code. It declares variables with names you provide and can even generate structures.

With VCScreen choosing the right functions, parameters and sequences, and Vitamin C supplying the functions to choose from, you can stop worrying about semi-colons, matching braces, and calling conventions and concentrate on creating your application!

### 30 Day **Money Back** Guarantee

Better than a brochure. More than a demo disk. If you're not satisfied, simply return the package within 30 days and receive a full refund of the purchase price.

#### Vitamin C ..... \$225.00

Includes ready to use libraries, tutorial, reference manual, demo, sample, and example programs, and quick reference card. For IBM PC and compatibles. Specify Microsoft, Lattice, Computer Innovations, Aztec, Mark Williams, Wizard, DeSmet, or Datalight C compiler AND compiler version number when ordering.

Vitamin C Source ... FREE\*

VCScreen ..... \$99.95

Requires Vitamin C and IBM PC/XT/AT or true compatible.

#### **ALL ORDERS:**

SHIPPING: \$3 ground, \$6 2-day air, \$20 overnight, \$30 overseas. Visa and Master Card accepted. All funds must be U.S. dollars drawn on a U.S. Bank. Texas residents add 61/8% sales tax.

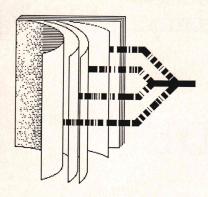
For Orders or More Information, Call ...

(214) 245-6090



Creative Programming Consultants, Inc. Box 112097 Carrollton, Texas 75011

## DDJ ON LINE



## PROLOG and the Future of AI

The following is an excerpt from a real-time conference held by Borland International on CompuServe on July 26, 1986. A complete transcript of this three-hour on-line conference on AI and PROLOG can be found in DL6 of the Borland SIG on CompuServe (<GO BOR100> KEYWORDS:CONFERENCE).

Larry Kraft, SYSOP of Borland SIG: Our panel of featured "speakers" to-day includes Borland's president, Philippe Kahn; assistant professor Mark Chignell of USC, and Mike Swaine, editor-in-chief of *Dr. Dobb's Journal of Software Tools*. The first part of this conference will consist of a panel discussion centered on numerous questions that were submitted in advance. Our first panelist to speak will be Mark Chignell.

Mark: I'll start with a little information on my background in AI and PROLOG. I am an assistant professor in the Department of Industrial and Systems Engineering at the University of Southern California. I have a Ph.D. in psychology and an M.S. in industrial and systems engineering. At USC I became interested in PROLOG as a practical implementation language for AI applications in engineering. My current research is concerned with the development of human-computer interfaces in engineering design and on-line information retrieval.

Here's the first question I'm going to answer: What is artificial intelligence? People usually point to smart computer programs and say, "That's AI." In the early days of AI, 1956—1970, AI was thought of as a process of domain-independent, general-purpose reasoning. More recently, people have focused on domainspecific knowledge and the kind of heuristic reasoning that experts use. Perhaps the main unifying feature of all AI applications is the element of machine reasoning. In vision, for instance, the program is reasoning about how to update its model of the visual environment based on the sensory data. In planning, the program is reasoning about how to act on its model of the task environment so that a set of goals can be achieved and so on. (See P. McCorduck, Machines Who Think, for a historical introduction to the issues faced by AI.)

**Philippe:** Well, to me AI is what hasn't been done yet—once it's been written, it's called programs!

Mark: Question: What are "true" AI applications? Perhaps the thing that distinguishes AI from other applications is the need for symbolic reasoning. In statistics, for instance, it wouldn't make much sense to use an AI program to find the straight line that had the best least-squares fit to a set of data. That task is already done well by numerical algorithms. In machine chess, brute-force methods based on grinding through possible move sequences do yield fairly good results, but the problem of combinatorial explosion of search possibilities has led to an examination of how human masters perform the task and has led to attempts to incorporate their knowledge representations and heuristics into chess programs.

**Philippe:** Well, that is one way of typing things.... On the other hand, I think that five years ago people would have called a resident, beeping spelling checker AI.

Mark: I guess we have a difference of opinion here. I think it should be possible to characterize AI independently from the current status of technology. We should be moving toward a definition of intelligence that covers both humans and machines.

Question: How can you tell if a program is intelligent? In today's cli-

mate, there is a tendency to assign the label AI rather liberally. Deciding on whether a program is intelligent is a particular case of the general problem of recognizing intelligent behavior. One method for establishing the intelligence of a program is a type of Turing test. If the performances of a program and of a person on a task that requires intelligence are virtually indistinguishable, then we assume the program is intelligent.

**Larry:** Thank you very much, Mark. Mike, you're up next. Go ahead, please.

Mike: I'm Mike Swaine, editor-inchief of *Dr. Dobb's Journal of Software Tools*. My background includes graduate study in both human cognition and artificial intelligence and three years reporting on AI and new technologies as a senior editor for *InfoWorld*. I am coauthor of *Fire in the Valley*, a history of the personal computer, and creator of the fictional puzzle-detective Mr. Usasi.

Question: What is declarative programming, and why would you want to use this type of programming? Declarative programming stresses static aspects of knowledge: facts about the world and rules about how the facts are connected. It concentrates on representing these facts and rules, and it deliberately submerges all procedural details. These procedural details are nothing less than the entire control structure of the program-that is, what statement gets executed next or, in more conceptual terms, how to use these static facts and rules to answer questions, solve problems, or derive new facts and rules.

PROLOG, for example, uses the model of first-order predicate logic to represent the facts and rules about some domain of knowledge—such as U.S. geography—and submerges the procedural details in an inference engine, a mechanism that automatically makes the necessary deductions from the facts and rules. To oversimplify, using PROLOG means pouring facts and rules into the system, asking questions, and letting the system derive the answers from the informa-

tion you have supplied. You declare; it deduces. To the extent that declarative programming actually submerges the procedural details, it achieves one of the goals of what is called fifth-generation language design: it allows the programmer to focus on the problem rather than on the program. In implementing a geographical database, for example, you can concentrate on facts about U.S. geography rather than on details of database design.

Question: What are the advantages and disadvantages of declarative vs. procedural programming? The choice of a declarative or a procedural approach to solving a particular problem can depend on what kind of knowledge about the problem is most accessible. If you can gather the important facts and rules about the problem domain easily, then you should consider a declarative approach. If it's easier to specify the steps or techniques for solving the problem, then you should consider a procedural approach. Another consideration is consistency vs. efficiency. A declarative, first-order, predicate-logic-based approach can be trusted to be consistent; you won't get false conclusions from true premises. But by giving up control over the way the program searches for solutions, you give up the option of fine-tuning the code for efficiency. A procedural approach lets you specify how to solve the problem efficiently but at the cost of introducing complexities that make it harder to trust the results.

Question: Are PROLOG and LISP both declarative? No language is strictly one or the other, although most programming languages are mainly procedural. LISP can be thought of as declarative, but it's a funny fit—LISP wants to be thought of as functional, and it's old enough to be humored. PROLOG was designed to be used declaratively. PROLOG probably gains in efficiency by not being purely declarative, but it pays for it in inconsistency because of extensions to the basic idea and, I think, to the way in which falsity is implemented.

Philippe: Well, LISP really manipu-

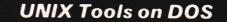
lates functions, and I would call it procedural, just like its contemporary FORTRAN.

Mike: OK. Next question: Why is PRO-LOG appropriate for writing expert systems, and what systems have been written in PROLOG? Last week I watched an expert systems "knowledge engineer" being grilled by a roomful of skeptical C programmers. The C programmers all wanted to know "What do you do that can't be done in C?" The knowledge engineer had to admit that anything he did could be done in C and that in fact his company typically ported its products to C for efficiency and portability. The programmers already knew these things, but it made them feel good to hear them.

So why use PROLOG for expert systems if you'll eventually rewrite them in C? Well, that's almost like asking why use a graphics language for graphics processing. Expert systems logically include certain compo-

nents that are built into PROLOG—like an inference engine; like a natural mechanism for adding to the knowledge base without rewriting the entire program. In fact, expert systems and PROLOG grew out of the same motivation: a desire to represent static knowledge in a computer program. Writing an expert system in PROLOG involves using some powerful tools. Rewriting it in C means recreating those tools. The latter may allow opportunities to optimize, but it also distracts attention from the real task.

As of today, though, PROLOG is not the language of choice for developing expert systems because of the past lack of a decent PROLOG programming environment. Of the hundreds of expert systems in nonacademic use in the U.S., nearly all were developed in some version of LISP, in a specialized expert-system-development language such as Teknowledge's S.1, or in a conventional thirdgeneration language such as C. (One



# MIKSTOOKit

#### HARVEST THE KORN

Over 70 programs bringing elements of UNIX System V.2 to the world of DOS. Our tools enhance your efficiency on machines like AT&T 6300, IBM PC, XT, AT and compatibles. We offer:

**shell** — Korn shell compatible — combines best features of Bourne & C shells

vi — a detailed implementation of the UNIX full-screen editor

awk — the only commercially-available version offering Bell Lab's latest published specs

cat	chmod	cmp	comm	ср	cpio	ctags	cut	date
dd	dev	df	diff	du	echo	ed	egrep	fgrep
file	find	head	help	join	lc	ls	more	mv
nm	od	paste	pg	prof	pwd	rm	sed	size
sort	split	strings	tail	time	touch	tr	uniq	wc
and m	uch much	more						

Programs come with complete UNIX-style command-line file name expansion and are not copy protected. Phone support 9-6 EST. Full documentation is included.

Price: \$139.00

#### Mortice Kern Systems Inc.

43 Bridgeport Rd. E., Waterloo, Ontario N2J 2J4

For information or ordering call collect:

(519) 884-2251

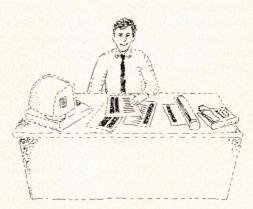
MasterCard & Visa orders accepted. OEM & dealer inquiries invited.
UNIX is a trademark of Bell Labs. MS-DOS is a trademark of Microsoft Corp

#### STREAMLINE YOUR PROGRAMMING

# SOFTSTRIP® NOW OFFERS YOU SOMETHING NEVER BEFORE AVAILABLE...



CONVENTIONAL DATA HANDLING



THE SOFTSTRIP SYSTEM

# A CHOICE.

Until now you were stuck with disks.

No more. Install our unique STRIPPER™ software on your personal computer today and discover the many benefits of the fastest, easiest, least expensive way to handle information.

STRIPPER lets you print – ON PAPER – your own machine readable Softstrip data strips using your dot matrix printer. The Softstrip System Reader reads that information into a computer rapidly. With STRIPPER and the reader, your PC and printer become part of the most versatile information handling system available.

With this system you can do anything you wish with any data you have in your PC – ON PAPER.

DATA ENTRY: Why use keystrokes when you can eliminate them with data strips? Whatever the document - invoices, packing slips, memos, letters, sales reports, the list is endless — simply print a data strip right on the same printed page. Now you have a document that is both human readable and machine readable. A typical document can be entered in only 15 seconds using data strips. And, it ends keystroke errors forever.

**DATA DISTRIBUTION: Why copy disks?** It's time consuming and expensive. Softstrip data strips will end all that. Simply photocopy as many data strips as you like and send them by mail. Data strips ignore folding, coffee stains, ink marks and, by the way,

magnetic fields. And if you're using telecommunications, you can stop making the phone company rich.

DATA STORAGE AND RETRIEVAL: Why have a file of disks and a file of paper? Eliminate one with Softstrip data strips. File the data strip with the document. Better still, print the strip right on the document. Then put it in a file or binder.

Retrieval is simple. To find existing data, pull the document and its related data strip from the file. They've been stored together. Then use the reader to enter the data. No more hassle trying to match documents with the right disk — if you can find it.

DATA TRANSFER: Why bother with cables, modems and phone lines to move files between computers? A Softstrip data strip generated by an IBM PC can be read into another PC, or compatible, an Apple or even a Macintosh. If you work at home on a Macintosh, make a data strip on your printer, take it into the office and read it into your IBM PC. Simple. And we've created the utilities to let you do that easily. (See Application Notes on opposite page.)

Fascinating, isn't it? Anything you can do with disks can be done with the Softstrip data strip system — faster, easier and at lower cost — ON PAPER.

All you need is STRIPPER software at \$19.95 and the Softstrip System Reader at \$199.95.



# NOW! TRANSFER DATA – PROGRAM TO PROGRAM WITH SOFTSTRIP®

Now you can move data between programs quickly and easily using SOFTSTRIP data strips.

Using the Softstrip System, you can move data between computers and such programs as WordStar and MacWrite, dBASE and AppleWorks, Lotus 1-2-3 and Excel and ReadySetGo and many others.

We've created a series of several dozen Application Notes on Softstrip data strips. These lead you through simple steps to make the file transfer as easy as possible, adding even more versatility to your personal computer when you purchase the SOFTSTRIP SYSTEM. The advanced system you've been hearing so much about.

All you need to move data between programs is STRIPPER™ software at \$19.95 and the Softstrip System Reader at \$199.95.

For a complete list of Application Notes, contact your dealer or call Cauzin.

## **JANUARY CASE HISTORY**



A comprehensive new book covering Turbo Pascal programming includes 15 pages of SOFTSTRIP data strips, enabling readers to input extensive programs using the SOFTSTRIP SYSTEM READER.

Jeff Duntemann, author of "Turbo Pascal, Second Edition — Revised and Enlarged" convinced publishers Scott, Foresman & Co., to include the data strips as a service to readers.

Some programs included in the book are lengthy — up to 800 lines. All 300K of listings are incorporated in the SOFTSTRIP data strips for easy entry with the Reader. An index indicates which files are contained in each strip.

Duntemann said inclusion of the data strips will let readers spend more time learning the intricacies of working with Turbo Pascal and significantly less time typing in the various programs.

Users' Groups: Call for Special User Group Discounts.

ACT NOW! Don't delay. See your local Softstrip dealer or call us at 1-800-533-7323. In Connecticut: 203-573-0150.

CAUZIN

835 South Main Street Waterbury, CT 06706 (203) 573-0150 For Europe and Asia Contact:

Softstrip International, Ltd.
53 Bedford Square
London, WC1 B3DP England
01-631-3775 Telex: 263874SOFTST G

This data strip contains IBM2MAC, a utility that runs on the IBM and converts an IBM file to Macintosh format.

Softstrip

Circle no. 307 on reader service card.

# Finally, powerful graphics for Turbo Pascal!

# Announcing: TurboHALO Graphics!

- 150 graphic primitives (based upon the powerful HALO primitives).
- Graphics in up to 16 colors, in medium or high resolution, on 10 popular graphics cards—including IBM CGA and EGA, Hercules, and AT&T DEB.
- Output to over 20 printers and plotters—including **HP** and Corona Laser and Apple Imagewriter.
- 11 popular input devices supported.
- Memory resident drivers require only 2K of Turbo Pascal code space.
- These professional graphics programs use HALO Primitives:

Drafix II — Foresight Resources
CADKEY — Micro Control Systems, Inc.
Dr. Halo II — Media Cybernetics/IMSI
CAD Master — Datagraphics
Decision Master — Centec
Artworks — West End Films

 Speed, ease of use, multiple fonts and much more, for ONLY \$129.00! NO ROYALTY FEES!

#### **COMPARE THE FEATURES!**

TurboHALO vs. Borland's Turbo Grafix ToolBox:

<b>Feature</b>	<b>TurboHALO</b>	Borland	
16-Color Support?	Yes	No!	
Full EGA Support?	Yes	No!	
Laser Printer Support?	Yes	No!	
Multiple Input Drivers?	Yes	No!	
Do <u>Professional</u> Programs Use Same Primitives?	Yes	No!	

## Call Now To Order!



1299 Fourth Street • San Rafael, California 94901 • (415) 454-7101 **Toll Free** (800) 222-GRAF (Outside CA) • (800) 562-GRAF (In CA)

Circle no. 293 on reader service card

#### **DDJ ON LINE**

(continued from page 131)

counterexample to keep you awake nights: Lockheed is using PROLOG to develop an expert system for the Department of Defense [DoD] to analyze electronic intelligence data to determine "enemy intentions." In Japan, PROLOG-based expert systems have been developed for (at least) medical, commercial, and engineering applications. The new PROLOG implementations coming to market may change this picture radically.

Philippe: Well, in Europe, where PROLOG was born, PROLOG has been more widely used than LISP. Furthermore, PROLOG is newer and younger, and it is just now picking up a lot of momentum.

Mark: The two other panelists classified LISP as a procedural language. As someone who has spent some time with the language, I feel some duty to defend it. LISP is really a language-development environment rather than a single language. Once you start writing functions, you can create your own language. LOGLISP is an example of a PROLOG-like language in LISP—that is, declarative versions of LISP have already been written. Object-oriented languages have also been written on top of LISP.

Mike: What does AI offer to the average programmer or user? I'll give only one of the answers; maybe others will emerge from discussion. AI is the domain of exploration of new programming techniques. When they cease to be new, they cease to be AI, but they don't cease to be useful. Also, I'd like to point out that PROLOG may be a good prototyping language for anything, not just AI applications.

I have a question. As editor-in-chief of a magazine for software developers, I am interested in the interface—in the sense of the zone of transmission—between the other two panelists' areas of expertise. I'm curious about developments in AI labs that may lead to commercial products in the future. I haven't been particularly prescient about this in the past. Having written a simple expert system in graduate school, I understood the principles. I had been following AI work closely when Teknowledge

was founded and knew the credentials of its founders; nevertheless, I did not foresee the current success of expert systems. Expert system companies are beloved of investment capitalists. Teknowledge was one of the few sales winners in a recent San Jose Mercury News summary of the sales slump in Silicon Valley. I'd like to do better the next time. I'd like to be able to see the next area of commercial development and practical application of laboratory developments in AI—the next Big Thing. There is a tantalizing suggestion of what that might be....

Mark: That is a very good point, Mike. I think it is often hard for researchers to predict what is going to fly in the marketplace. If I were to make a bet, I would say that in the near term we may see a revolution on retrieval and utilization of information/knowledge using AI-based front ends.

Larry: OK. Philippe's turn for guestions. Philippe, I believe you have some opening comments? Go ahead.

Philippe: First: My updated biography: Failed musician, mathematician, relatively artificially intelligent, self-appointed "the software industry's resident court jester"! Pops up unexpectedly anywhere.

Larry: Now, I have the questions for Philippe. What plans are there to tie PROLOG to conventional databases? Is this a growing area of AI technology?

Philippe: I don't know whether it's a growing area, but it should be very useful. The biggest problem people have with large databases, or in the AI world "knowledge bases," is reentering data. The best thing is to be able to read and write "usual" database files

Larry: Next question: Is PROLOG a general programming language that can be used for a wide variety of programming applications or is it specifically database-oriented?

Philippe: Well, it is inference-oriented, if anything. With good extensions, PROLOG can let you do different general things, but as with any tool, you need to use the right tool for the right job. If you use a hammer when you were supposed to use a saw, you might get into trouble!

> Alis the domain of exploration of new programming techniques.

Mark: You know, there are close to two billion documents on-line in the world today. This is a huge amount of information, but it's not really knowledge until you can distill the essential meaning. Perhaps the next big AI industry will be the replacement of much of the current knowledge-engineering effort with what I will call a "knowledge mining" effort, looking to translate the current backlog of electronic information into usable knowledge bases.

Philippe: There is much more in a lot of this information—things as simple as typesetting codes, tables of contents, indexes, cross-references, and so on. Millions of man-hours of editing have gone into that stuff. It is much more than dumb data, at least in many cases. You still have to interpret it, but a lot of the work has already been done. Take a book such as Roget's Thesaurus, for example. It divides the world into categories, and you can thus define a vector space in a given metric and talk of a much broader way to index data semantically rather than through a keyword system. But as our old friend Kipling said, "This is yet another story!"

DDJ

Vote for your favorite feature/article. Circle Reader Service No. 9.

#### The Advanced Programmer's Editor That Doesn't Waste Your Time

# EPS LO

- Fast, EMACS-style commands—completely reconfigurable
- Run other programs without stopping Epsilon—concurrently!
- C Language support—fix errors while your compiler runs
- Powerful extension language
- Multiple windows, files
- Unlimited file size, line length Supports large displays
- 30 day money-back quarantee Not copy protected
- Great on-line help system
- Regular Expression search

**Only \$195** 



5740 Darlington Road Pittsburgh, PA 15217



for IBM PC/XT/AT's or compatibles

Circle no. 135 on reader service card

# The fastest C

Your search for execution speed is over. The new Microsoft®C Compiler Version 4.0 is here. With blazing performance. We've added common sub-expression elimination to our optimizer that produces code that rips through the benchmarks faster than ever before.

"...the Microsoft performance in the benchmarks for program execution is the best of the lot overall." -William Hunt, PC Tech Journal, January, 1986\*

But speed isn't the only edge you get with Microsoft C. Other advantages include a variety of memory models like our new HUGE model that breaks the 64K limit on single data items. Plus our NEAR, FAR and HUGE pointers, which provide you greater flexibility. All this allows you to fine tune your program to be as small and fast as possible.

"Excellent execution times, the fastest register sieve, and the best documentation in this review ... Microsoft Corporation has produced a tremendously useful compiler."-Christopher Skelly, Computer Language, February, 1986.

> No more debugging hassles. Introducing CodeView. Free.

Now, for a limited time, we'll give you an unprecedented programming tool when you buy Microsoft C, free. New Microsoft Code-View<sup>™</sup> offers the most powerful tool yet in



the war on C bugs. Forget the hex dumps. Now you can view and work with programs at any level you want. Use the program source, the disassembled object code, or

#### Microsoft C Compiler Version 4.00

#### Microsoft C Compiler

- Produces fast executables and optimized code including elimination of common sub-expressions.
- · Implements register variables.
- Small, Medium and Large Memory model libraries.
- Compact and HUGE memory model libraries.

  Can mix models with NEAR, FAR and the new HUGE pointers.

  Transport source and object code between MS-DOS\* and XENIX\*
- operating systems.
- Departing systems.
   Library routines implement most of UNIX™ System V C library.
   Start up course code to help create ROMable code.
   NEW! · Start-up source code to help create ROMable code.
- Full proposed ANSI C library support (except clock) NEW!
- Large number of third party support libraries available.
  Choose from three math libraries and generate in-line 8087/80287
- instructions or floating point calls:
  - floating point emulator (utilizes 8087/80287 if installed).
  - 8087/80287 coprocessor support. alternate math package - extra speed without an 8087/80287.
- Link your C routines with Microsoft FORTRAN (version 3.3 or higher), Microsoft Pascal (version 3.3 or higher) or Microsoft Macro Assembler.
- Microsoft Windows support and MS-DOS 3.1 networking support.
   Supports MS-DOS pathnames and input/output redirection.

#### Microsoft Program Maintenance Utility. NEW!

- Rebuilds your applications after your source files have changed.
- Supports macro definitions and inference rules.

#### Other Utilities

- · Library Manager.
- Object Code Linker.
- EXE File Compression Utility.
- EXE File Header Utility.

#### C Benchmarks

#### In seconds

	Microsoft C 4.0	Lattice C 3.0	Computer Innovation C 2.3	Aztec C86 3.2	Wizard C 3.0
Sieve of					
Eratosthenes (register)	82.9	151.4	172.3	88.0	91.9
Copy Block	86.9	231.7	199.0	123.8	189.5

Run on an IBM PC XT with 512K memory

#### Microsoft CodeView Window-oriented source-level debugger. NEW!

- Watch the values of your local and global variables and expressions as you debug.
- Set conditional breakpoints on variables, expressions or memory; trace and single step.
- · Watch CPU registers and flags as you execute
- · Effectively uses up to four windows
- Debug using your original source code, the resulting disassembly or both intermingled.
- · Use drop-down menus to execute CodeView commands
- · Access the on-line help to lead you through CodeView's options and settings.
- Easily debug graphics-oriented programs since program output is kept separate from debugger output.
- Keyboard or optional mouse support
- Enter in familiar SYMDEB or DEBUG commands.

# you've ever seen.

both at the same time. Open a window to view CPU registers and flags. Watch local and global variables as well. All while your

program is running.

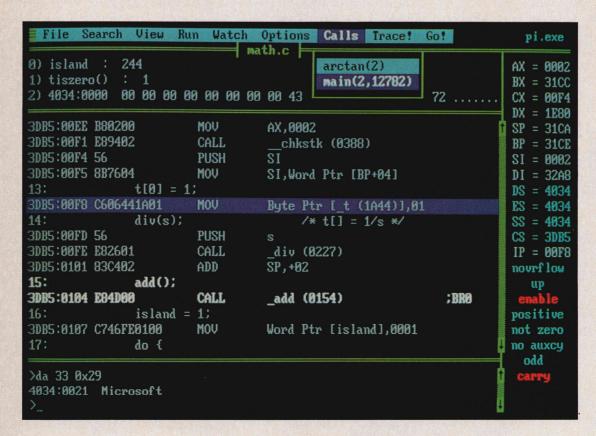
CodeView gives you complete control. Trace execution a line at a time—using source or assembly code. Or set conditional breakpoints on variables, memory or expressions. CodeView supports the familiar SYMDEB command syntax, as you'd expect. Commands are also available through dropdown menus. Combine the new window-oriented interface with our on-line help and debugging has never been easier. Or quicker.

Take the \$5 CodeView tour.

You may find it hard to believe our debugger can do all we've claimed. So we're offering test drives. Five bucks will put you behind the wheel of a Microsoft C demo disk with CodeView.† See for yourself how fast debugging a second of the code of

ging can get.

For more information about the Code-View demo disk, the new Microsoft C Compiler, a list of third party library support or the name of your nearest Microsoft dealer, call (800) 426-9400. In Washington State and Alaska, (206) 882-8088. In Canada call (416) 673-7638.



## Microsoft<sup>®</sup> C Compiler

The High Performance Software

(continued from page 14)

indeed a principle of classical logic. As such it does not support the former invalid inference. The assumption PROLOG makes is the altogether different assumption termed the closedworld assumption. PROLOG automatically assumes that any postulate set (knowledge base) is complete: if it cannot be derived that S, then it must be that not S. Thus PROLOG enshrines the fallacy dubbed by Spinoza as the argumentum ad ignorantiam: If it can't be proven true, it's false, and if it can't be proven false, it's true. Of course it follows that a proposition that can't be proven true or false is both true and false. Indeed, with the proper repositioning of postulates (rules), PROLOG will answer "yes" and "no" to the same query even though the postulate set is consistent.

The point is that if a system is not complete (there are such complete systems—real closed fields being an example), then the assumption of completeness made by PROLOG (built into its definition of negation) is false and will lead to fallacious inferences

and contradictory inferences. The justification offered for PROLOG's treatment of negation is that *not* means *not known* or *not derivable*. But this lame attempt at justification doesn't hold up. Neither the epistemic nor the apodictic concept obey DeMorgan's laws, whereas the truthfunctional *not* in PROLOG does.

It is simply not safe to use not unless it is pinned down to a range (for example, with the use of ON). Otherwise the negation logic needed should be provided by the programmer. Negated sentences can be treated as unitsfor example, use of not-L instead of not L. The relationship between L and not-L and other negation relationships must be spelled out by the programmer. The programmer must use (either not-L or not-K) instead of not(L and K) and so forth. PROLOG never actually transforms any of the rules in the knowledge base, which means that the programmer can provide the negation logic needed.

Texts and manuals for PROLOG should be up front about PROLOG's

limitations. It is not a full predicate logic in any direct sense. What PRO-LOG is is a negation-restricted, expanded logic of definition with marvelous recursive powers. Properly billed, the foregoing facts about PRO-LOG's inconsistency and radical incompleteness merely become irrelevant considerations based on a confusion about what PROLOG is supposed to be. Consider the very first problem with the introduction of a postulate declaring the transitivity of R. Considered as a definition, the postulate violates the cannons of definition by attempting to define R nonrecursively in terms of itself. PROLOG can easily handle the introduction of a transitive relation when defined recursively. The transitive closure TR of a relation R is defined:

TR(x y) if R(x y)TR(x z) if R(x y) and TR(y z)

The general theory of definition and the theory of recursive definition can be given a rigorous syntactical formulation—it would be interesting to see an exact syntactical formulation of the extension used by PROLOG. From a computer science point of view, this would amount to giving syntactical rules to rule out non-logic-based semantic errors (logic errors in the field of partial recursive functions cannot be ruled out by syntactical rules).

The deductive-axiomatic method has been the central unifying methodology of knowledge of the Western intellectual tradition. By removing the barrier to the real-time use of the deductive-axiomatic method, PROLOG may have an impact on knowledge use and acquisition that is hard to overestimate. After all, being able to query Aristotle, or Goethe, or Einstein with an updated database does give new meaning to the expression deus ex machina.

Those who package PROLOG should have the courtesy and integrity to say what it is and what it isn't.

DDJ

Vote for your favorite feature/article. Circle Reader Service No. 1.

## PL/I FOR PROS

Your existing PL/I applications *can* run on today's hot new workstations.

Because LPI-PL/I runs under UNIX, you can move programs simply by recompiling—without re-programming. And, because it's a true high-performance compiler, LPI-PL/I makes full use of the speed of those workstations.

For more information, contact Language Processors, Inc., 400-1 Totten Pond Rd., Waltham, MA 02154 (617) 890-1155.



LPI-RPGII, LPI-COBOL, LPI-PL/I, LPI-BASIC, LPI-FORTRAN, LPI-PASCAL, LPI-C, LPI-DEBUG LPI is a trademark of Language Processors, Inc. UNIX is a trademark of AT&T.

## IS GETTING THE ANSWER TO SOFTWARE PROBLEMS A BIGGER PROBLEM THAN THE PROBLEM?

Don't stay on hold when there's help online from CompuServe<sup>®</sup> Software Forums.

The new upgraded version of your software locks up.
And every time you reboot,

you get stuck in the same place in the program.

You've chucked the manual, because you've done exactly what it tells you to do six times already. So you call the software company.

Now you spend half a day beating your head against a brick wall of busy signals, ranting at recorded messages, hanging around on hold. And you still don't get the solution to your problem.

Meanwhile, progress is stopped

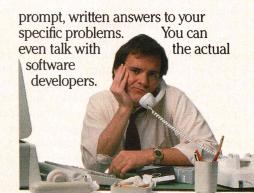
and your profits are dribbling away. But wait. There's help...

Several prominent, progressive software publishers recognize this problem, and working with CompuServe, have developed a solution—

CompuServe Software Forums.

Now you can go online with

Now you can go online with experts from the companies that produced your software and get



Adobe Systems, Aldus, Ashton-Tate, Autodesk, Borland International, Creative Solutions, Digital Research, Living Videotext, Lotus Inc., Microsoft, MicroPro, Misosys Inc. and Software Publishing all have CompuServe Software Forums.



CompuServe's large subscriber base also puts you in touch with thousands of other, often more experienced, users of the same software. You'll find they can give you lots of creative ways to get the most out of your software.

And software forums are the best way to learn about product updates, new product announcements, new ways to expand the uses of your software, and offer free uploads of your own programs.

Our online electronic magazines

frequently publish software reviews. And you can find help for many other software products in our other computer-related forums for IBM, Tandy, Atari, Apple, Commodore, TI and others.

The last thing you need when you've got a software problem is a bigger problem getting answers. So, from now on, get prompt, informed answers on CompuServe Software Forums.

To buy your CompuServe Subscription Kit, see your nearest computer dealer. Suggested retail price is \$39.95.

To order direct or for more information, call 800-848-8199 (in Ohio, 614-457-0802).

If you're already a CompuServe subscriber, just type GO SOFTWARE at any! prompt.



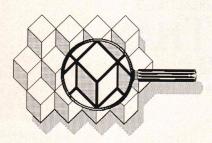
### CompuServe<sup>®</sup>

Information Services, P.O. Box 20212 5000 Arlington Centre Blvd., Columbus, OH 43220

An H&R Block Company

Circle no. 237 on reader service card.

#### OF INTEREST



#### **Artificial Intelligence**

ESP Frame-Engine is an expert system shell from Expert Systems International. Designed for customized enhancements, the shell uses frames that allow for rule-based and variable groupings as well as inheritance. Object-oriented programming is possible with several types of objects such as numeric, Boolean, text, set, and instance. The shell's open-ended architecture facilitates interfacing to Prolog-2, C, and other languages. Reader Service No. 16.

Expert Systems International

Expert Systems International 1700 Walnut St. Philadelphia, PA 19103 (215) 735-8510

The second edition of Experiments in Artificial Intelligence for Microcomputers by John Krutch has been published by Howard W. Sams & Co. This edition contains 75 percent more material providing step-by-step procedures detailing how AI can be applied to a variety of practical activities. Programs are provided in BASIC for the Commodore 64 and 128, with instructions for converting them to other BASICs. Reader Service No. 17. Howard W. Sams & Co. 4300 W. 62nd St. Indianapolis, IN 46268 (800) 428-SAMS

Borland International has released an enhanced version of Turbo Prolog that provides more support for the development of large applications. The new version (1.1) has a faster compilation speed and an internal linker, with single-step compiling to executable files. It also requires less space in memory than the previous version. It costs \$99.95 (free to registered owners of Version 1.0). Reader Service No. 18. Borland International 4585 Scotts Valley Dr. Scotts Valley, CA 95066 (408) 438-8400

#### Languages

CET Technology has released CET BA-SIC, a compiled application development language for Intel-based Unix and Xenix systems. The CET BASIC compiler is compatible with OASIS, THEOS, and UX-BASIC, and CET BASIC programs can be intermixed with programs and subroutines in other languages. Additional features include multiuser support for ISAM, direct and sequential files, terminal independence, error trapping, program chaining, and COBOL-like formatting. The CET BASIC compiler is available for \$695. Reader Service No. 19. CET Technology Inc.

5405 Garden Grove Blvd., Ste. 160 Westminster, CA 92683 (714) 895-4345

Rational Systems has released Instant C, an incremental C compiler that pares down development time by processing only those parts of a program that have been changed. The compiler incorporates a full-screen editor; source-level debugger; object code linker; source code checker; run-time checker; and support for linking Lattice C, Versions 2.0 and 3.0, and Microsoft C, Version 3.0, object code and libraries. It runs on computers with MS-DOS or Concurrent DOS and costs \$495. Reader Service No. 20.

Rational Systems Inc. P.O. Box 480 Natick, MA 01760 (617) 653-6194

Lattice now offers the SSP/PC library of more than 145 mathematical subroutines for use in scientific, engineering, and statistical computations. The subroutines can be called from Lattice C and provide PC programmers with routines similar to packages used on mainframes. Most routines yield a maximum machine accuracy of 15 significant figures. The SSP/PC library sells for \$350.

Reader Service No. 21. Lattice Inc. P.O. Box 3072 Glen Ellyn, IL 60138 (312) 858-7950

Marshal Pascal from Marshal Language Systems is a code-optimized ISO Pascal compiler for MS-DOS, CP/M-86, and Concurrent DOS. The compiler lets you address as much memory as your operating system allows, and it supports a variety of memory models. Marshal Pascal costs \$189. Reader Service No. 22.

Marshal Language Systems 1136-P Saranap Ave. Walnut Creek, CA 94595 (415) 947-1000

Lifeboat Associates has introduced a C++ language for micros. Advantage C++ provides extensions and enhancements, using C++ as a preprocessor to emit pure C code. C++ allows you to design your own data types and enables you to use object-oriented programming methods. Versions of Advantage C++ are available for use with Lattice C, Microsoft C, and other popular C compilers for \$495. Reader Service No. 23. Lifeboat Associates Inc.

55 South Broadway Tarrytown, NY 10591 (914) 332-1875

The Whitewater Group has released a new object-oriented programming language that incorporates Microsoft Windows. ACTOR is a fast and powerful programming environment that uses Pascal-style syntax. DDE (Dynamic Data Exchange) has been implemented for Microsoft Windows to allow for the simultaneous transfer of information between separate programs on the same computer. ACTOR runs on the IBM PC, PC/XT, or PC/AT and sells for \$495. Reader Service No. 24.

The Whitewater Group Technology Innovation Center 906 University Pl. Evanston, IL 60201 (312) 491-2370

**Workman & Associates** has released FTL Modula-2 for MS-DOS. The software provides a complete language

## WE JUST GOT MORE SOPHISTICATED SO YOU CAN GET MORE BASIC.

e invented BASIC over 20 years ago.
Later, we re-invented it for micros as the True BASIC™ structured-programming language.

And the idea was: To make programming as easy and natural as possible. So you could concentrate on what to program. Not how.

Now there's True BASIC Version 2.0 for the IBM® PC and compatibles. Faster, more powerful and sophisticated than the original.

#### **MORE GRAPHICS.**

Right from the start, True Basic gave you terrific device-independent graphics. Built-in 2-D transforms. And support for multiple windows.

Now we've added more graphics and full mouse support.

So for the first time, you can create one program that will do superb graphics on CGA, EGA or Hercules displays. Without worrying about additional drivers or overlays. And on the EGA, you can SET COLOR MIX to define your own colors. Use four shades of blue if you want (and make our competitors green with envy).

#### MORE CONTROL.

We always supported you with recursion, local and global variables and separately compiled libraries.

Now you can have *modules*, too, the industrial-strength tool for building large applications.

Using modules makes it easier for you to share data between routines. Build data structures. Then, if you want, hide them from other parts of the program. So you can always be free to focus on the task at-hand.

Modules have their own initialization sections, so you can set up global variables or turn on instrumentation.

And, like other procedures in True

BASIC, modules can be compiled separately and stored in a library where they can be shared by several applications. Or they can be loaded directly into the True BASIC environment as part of your customized workspace. So when you use True BASIC interactively, the modules look like built-in functions.

Modules made Modula-2 the successor to Pascal. Now they've put True BASIC one-up on all other BASICs.

#### MORE SPEED.

2.0 is 20 to 200 percent faster than True BASIC Version 1.0. Both compile times and execution speeds. And on some real-world benchmarks, we're faster than many native-code compilers.

#### MORE POWER.

Start with a complete matrix algebra package.

Then, since we support the use of 640K for both code and data, add arrays as large as you want.

Our compiled code is more compact than what other compilers generate, so there's more memory left for your application.

We've enhanced our dynamic array redimensioning and improved our built-in 8087/80287 support, making True BASIC the most powerful number-crunching BASIC around.

And if it's strings you crunch, we've added new string functions and raised the limit. So strings can be up to 64K characters long.

#### MORE DEBUGGING.

We pioneered breakpoints and immediate-mode capability in a compiled BASIC environment.

Now we've added utilities that allow you to visually TRACE through your program, and check the values of selected variables. Or print a crossreferenced listing.

Circle no. 344 on reader service card.

And new compiler options like NO LET and NO TYPO let you decide how strictly you want your variable names checked.

#### MORE INNOVATION.

True BASIC has always had features like full-screen, scrollable editing. Block copy and block moves. And global search and replace.

Now, 2.0 keeps you on the leading edge of editing and file-management technology. With SCRIPT, to write the True BASIC equivalent of a DOS batch file. ECHO, to transfer your output to disk or printer. And ALIAS, to give you and your programs a better roadmap to your subdirectories.

There's also Version 2.0 of the Developer's Toolkit. With support for DOS interrupts. Pop-up menus. Even designer fonts.

And remember: your programs are portable to the other machines we support: the Apple Macintosh™ and Commodore Amiga.®

#### MORE SUPPORT.

Call your local dealer. Call us TOLL-FREE at 1-800-TR-BASIC. Or write to: True BASIC, Inc., 39 South Main Street, Hanover, NH 03755. We'll send you more information. Including a free demo disk.

See for yourself. That we're still true to our basic idea.



True BASIC Language System is a trademark of True Basic, Inc. Macintosh is a trademark licensed to Apple Computer Inc. Amiga is a registered trademark of Commodore-Amiga, Inc. IBM is a registered trademark of International Business Machines.

### THE ULTIMATE IN COMPACT CP/M COMPATIBLE COMPUTERS

#### DSB-8100

#### Features:

- Hitachi 64180
   CPU running at 6MHz
   (executes a superset of Z80 instruction set)
- 256K dynamic RAM
- 8K EPROM with boot 7 monitor program standard, up to 32K EPROM optional
- 1773 Floppy controller supports 40 and 80 track 51/4" and 31/2" drives
- Host / target SCSI port can use DMA for all transfers
- Two RS-232 serial ports support asynchronous communications up to 38,400 baud
- · Centronics type parallel printer port
- CP/M 2.2 optional
- Power requirements:
- +5V at 1.0A
- + 12V at .05A
- Size: 6-3/4" x 3-7/8"

\$365.00 Quantity discounts

Compatible board with 512K RAM and 6 serial ports also available.

Davidge Corporation P.O. Box 1896 94 Commerce Drive Buellton, CA 93427 (805) 688-9598



Circle no. 353 on reader service card.

#### SCIENTIFIC/ENGINEERING GRAPHICS TOOLS

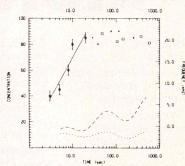
for the IBM PC

FORTRAN/Pascal tools: **GRAFMATIC** (screen graphics) and **PLOTMATIC** (pen plotter driver)

These packages provide 2D and 3D plotting capabilities for programmers writing in a variety of FORTRAN/Pascal environments. We support MS, R-M and IBM FORTRAN and more. PLOTMATIC supports HP or Houston Instrument plotters.

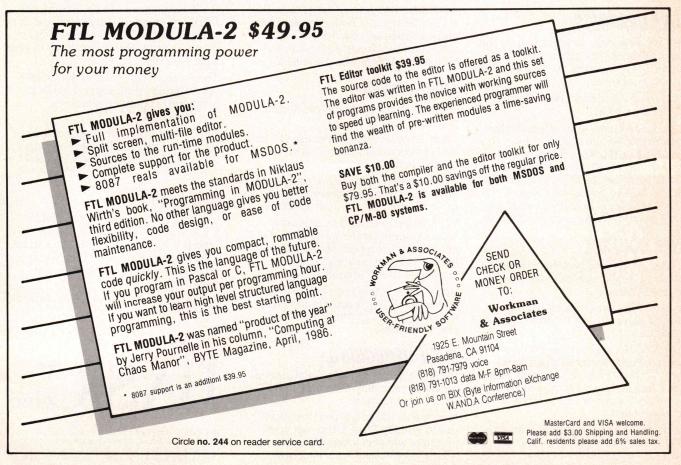
**Don't want to program?** Just ask for **OMNIPLOT!** Menudriven, fully documented integrated scientific graphics. Write or call for complete information and ordering instructions.

GRAFMATIC - PLOTMATIC - OMNIPLOT [S] & [P]



Microcompatibles, 301 Prelude Drive, Silver Spring, MD 20901

Circle no. 286 on reader service card



(continued from page 141)

development system, including compiler, linker, integral editor, library source, assembler, and support. MS-DOS libraries include "peek and poke" throughout DOS memory, low-level DOS calls, and a debugger. The company also sells the editor's source code, which is written almost entirely in Modula-2, for \$39.95. FTL Modula-2 costs \$49.95. A package of FTL Modula-2 and the editor's source code costs \$79.95. Reader Service No. 25.

Workman & Associates 1925 East Mountain St. Pasadena, CA 91104 (818) 791-7979

#### Tools

PC/Assembler from Computer Systems Documentation is an interactive syntax-checking assembler for the Intel 80xx, 801xx, and 802xx and the NEC V20/30 processors. PC/Assembler is used to write assembler subprograms that can be invoked from a high-level language. It is not copy-protected and costs \$99. Reader Service No. 26.

Computer Systems Documentation P.O. Box 5478 Albuquerque, NM 87115

TurboMAGIC, a code generator for Turbo Pascal programmers, is now available from **Sophisticated Software**. TurboMAGIC includes a full-featured editor and the ability to create both pop-up and pull-down menu systems. The form image can be stored either as a typed constant or in a picture file. The software runs on the IBM, PC/XT, PC/AT, or compatible computers with 256K and is not copy-protected. It costs \$99. Reader Service No. 27.

Sophisticated Software 6586 Old Shell Rd. Mobile, AL 36608 (205) 342-7026

#### **Expanding the IBM PC**

Fort's Software has released NVRD, the Non-Volatile RAM-Disk, a software package designed to work with expanded memory hardware or with the company's Virtual Expanded Memory Manager (V-EMM). Combined with V-EMM hardware, NVRD provides the improved performance of a RAM disk with the nonvolatility

of a hard disk. It runs on IBM PCs and compatibles with DOS 2.0—3.2, 192K RAM, a fixed-disk drive and fixed-disk adapter, and an EMS or V-EMM board. NVRD is available on its own for \$49.95 or bundled with the V-EMM for \$119.90. Reader Service No. 28.

Fort's Software P.O. Box 396 Manhatten, KS 66502 (913) 537-2897

MotherCard 5.0, a plug-in card for IBM PC/XTs and compatibles that offers full AT-compatibility with all software written for the 80286, in-

cluding protected mode operating systems. The on-board 1-megabyte RAM is all usable, and a DaughterCard connector is included that allows memory expansion to 16 megabytes. MotherCard 5.0 sells for \$995. Reader Service No. 29.

SOTA Technology 657 N. Pastoria Blvd. Sunnyvale, CA 94086 (408) 245-3366

An 8-megabyte memory expansion board for the IBM RT/PC is available from **Tall Tree Systems.** The JRAM-RT is a 32-bit board that makes use of the host motherboard's hardware to



#### FEATURES:

- ► Optional strong type checking
- Overloading of function names and operators
- Optional guaranteed initialization of data structures
- ► Data abstraction
- ► Dynamic typing (virtual functions)
- Optional user-defined implicit type conversion

The only commercially available C++ customized to operate on PC's, micros, minis, and mainframes with popular C compilers, including:

VAX C LATTICE MICROSOFT GREEN HILLS WIZARD WHITESMITH'S

- Works with your present C Compiler
- Functions as a Preprocessor Translator handles regular C code with no changes
- Type-checking and other features are optional you can turn them off
- Already thousands of users at commercial sites
- Complete documentation: C++, A User's Guide by Bjarne Stroustrup of AT&T (Addison-Wesley, 1986)

We Specialize In: Cross/Native Compilers: C, Pascal, FORTRAN, Ada, LISP—Assemblers/Linkers — Symbolic Debuggers — Simulators — Interpreters — Profilers — QA Tools — Design Tools — Comm. Tools, — OS Kernels — Editors — VAX & PC Attached Processors and more We Support: 680xx, 80x86, 320xx, 68xx, 80xx; Clipper, and dozens more

A DIVISION OF XEL

60 Aberdeen Ave., Cambridge, MA 02138 (617) 491-4180

er C++is a joint trademark of XEL, Inc. and Glockenspiel, Ltd of Dublin, Ada is a trademark of the U.S. Government (AJPO)

Circle no. 254 on reader service card.

#### OF INTEREST

(continued from page 143)

find and correct memory errors. It sells for \$3,995. Reader Service No. 30. Tall Tree Systems 1120 San Antonio Rd. Palo Alto, CA 94303 (415) 964-1980

Micro Enhancer from Everex Systems is a 5-inch short card that makes EGA capabilities available for users with limited space in their IBM PC/XTs or compatibles. The board provides 640 × 350-pixel resolution graphics in 16 colors from a palette of 64 colors and is 100 percent compatible with the IBM Enhanced Graphics Adapter. To simplify using the board, Everex also supplies its EGMODE menu-driven software. Micro Enhancer costs \$499. Reader Service No. 31.

Everex Systems Inc. 48431 Mimont Dr. Fremont, CA 94538 (415) 498-1111

**Personal Computer Support Group**'s half-slot speed-up board called the Breakthru 286 replaces the CPU of an IBM PC or PC/XT with an

80286 microprocessor faster than the one found in the 6-MHz IBM PC/AT. PCSG claims that the Breakthru 286 can beat the performance of other caching speed-up boards and that better performance can be expected in nearly all applications. The Breakthru 286 costs \$595. Reader Service No. 32.

Personal Computer Support Group 11035 Harry Hines Blvd., #207 Dallas, TX 75229 (214) 351-0564

#### For the Mac

Datacopy Corp. has released two scanning systems—the Jet Reader and the Model 730—that produce high-resolution images containing 300 dots per square inch. Once scanned, images can be formatted for insertion into documents produced with a Macintosh desktop-publishing program. The company's MacImage software lets you control the scanner and manage, print, and view image files. The JetReader with MacImage software is priced at \$2,250; the Model 730 sells for \$3,250. Reader Service

No. 33. Datacopy Corp. 1215 Terra Bella Ave. Mountain View, CA 94043 (415) 965-7900

THINK Technologies has introduced Lightspeed Pascal, a full ANSI Pascal. Lightspeed Pascal supports the Macintosh Toolbox and operating system and Apple's SANE extended numerics software. Compatible with Macintosh Pascal and Lisa Pascal, it runs on Macintosh computers with 512K RAM or more and is not copy-protected. It costs \$125. Reader Service No. 34. THINK Technologies Inc.

420 Bedford St. Lexington, MA 02173 (617) 863-5590

#### Miscellaneous

Dynapro Systems has announced chronOS, a real-time multitasking operating system that lets you use standard DOS programming tools to write real-time applications. Source code is included for the console and sound generator drivers, language in-

# Changing Your Address? We'd Like to Know.

To change your address, attach your address label from the cover of the magazine to this coupon and indicate your new address below.

	affix label h	ere
		I
Name		
Address		Apt. #
City	State	Zip
Dr. Dobb's Jou	Mail To:	7809, San Diego, CA

92128

#### C-PROGRAMMERS File System Utility Libraries All products are written entirely in K&R C. Source code included, No Royalties, Powerful & Portable. BTree Library High speed random and sequential access. Multiple keys per data file with up to 16 million records per file Duplicate keys, variable length data records 40.00 ISAM Driver Greatly speeds application development. Combines ease of use of database manager with flexibility of program-Supports multi key files and dynamic index definition. Very easy to use 59.00 Make Patterned after the UNIX utility. Works for programs written in every language. Full macros, File name expansion and built in rules. Full Documentation and Example Programs Included. ALL THREE PRODUCTS FOR -For more information call or write 1343 Stanbury Drive Oakville, Ontario, Canada L6L 2J5 (416) 825-0903 Credit cards accepted. Dealer inquiries invited.

#### PC/VI

#### Full Screen Editor for MS-DOS (PC-DOS)

Looking for an Ultra-Powerful Full-Screen editor for your MS-DOS or PC-DOS system? Are you looking for an editor FULLY COMPATIBLE with the UNIX\* VI editor. Are you looking for an editor which not only runs on IBM-PC's and compatibles, but ANY MS-DOS system? Are you looking for an editor which provides power and flexibility for both programming and text editing? If you are. then look no further because PC/VI IS HERE!

The following is only a hint of the power behind PC/VI: English-like syntax is command mode, mnemonic control sequences in visual mode; full undo capability; deletions, changes and cursor positioning on character, word, line, sentence, paragraph or global basis; editing of files larger than available memory; powerful pattern matching capa-bility for searches and substitutions; location marking; joining multiple lines; auto-indentation; word abbreviations and MUCH, MUCH MORE!

The PC/VI editor is available for IBM-PC's and generic MS-DOS based systems for only \$149. For more information call or write:

Custom Software Systems P.O. Box 678 Natick, MA 01760 617-653-2555

The UNIX community has been using the VI editor for years. Now you can run an implementation of the same editor under MS-DOS. Don't miss out on the power of

\*UNIX is a trademark of AT&T Bell Laboratories.

Circle no. 268 on reader service card

#### **HEXTOOLS**

Hex File Utilities for the Software/Hardware Developer

HEXTOOLS™ assists you in:

- ☐ Hex file offsets & partitioning
- ☐ Hex file concatenation
- □ Optional fill data
- ☐ Change load & execute address
- ☐ ROM/EPROM checksums

HEXTOOLS solves the hex file dilemma experienced by most developers.

Hex file too long or too short? Wrong load address? Need special fill data or a checksum at a specific location in a ROM or EPROM? HEXTOOLS can handle it all and more.

Available in INTEL hex format — PCDOS/MSDOS Other formats available soon

- \$49.95 -

... LEADERS IN PRODUCTIVITY ...



Research orporation

4 Townsend West-Suite #3 Nashua, NH 03063

603-880-4000

Circle no. 341 on reader service card.

## **IOCLISP**

More Common Lisp features in less space for less money than any other IBM-PC lisp.

- MSDOS portable
- Bignums, 8087 support
- Multidimensional arrays
- Full Common Lisp package system
- Full set of control primitives.
- Keyword parameters, macros
- Save/restore full environments for speed
- STEP, TRACE, BREAK, DEBUG, ADVISE. **APROPOS**
- Roll-out frees space for invoking MSDOS commands

**IQCLISP PACKAGE \$300.** 



q Integral Quality P.O. Box 31970 Seattle, Washington 98103 (206) 527-2918

Now with a compiler.

- Compiler comes with source
- Compiler cuts execution time 50-75%, program size 60-80%
- Multidimensional arrays
- Floating point, bignums, 8087 support
- Macros
- Color graphics
- Multiple display windows
- Assembly language interface
- Available for IBM PC or TI-PRO

**IQLISP PACKAGE \$270. INCLUDES COMPILER** 

- VISA and Mastercard accepted
- Generous update policy
- Attractive educational license

Circle no. 327 on reader service card.

(continued from page 144)

terfaces, and demo program. ChronOS runs on the IBM PC/XT or PC/AT and is tailored specifically for the iAPX86 microprocessor line. A U.S. site license costs \$1,995, and a Canadian site license costs \$2,495. Reader Service No. 35.

Dynapro Systems Inc. 1000 – 1200 W. 73rd Ave. Vancouver, B.C. Canada (604) 263-2638

**Alligator Transforms** has released Prime Factor FFT, a fast Fourier transform subroutine library for the IBM

PC, PC/XT, PC/AT, and compatibles equipped with an 8087 math coprocessor. The library can be called from any high-level language, and interface examples are provided in all languages. The library includes forward and inverse FFTs for single- and double-precision, floating-point, complex number sets. Users are not limited to radix2 data set sizes Prime Factor FFT sells for \$159.

Reader Service No. 36. Alligator Transforms P.O. Box 11386 Costa Mesa, CA 92627 (714) 662-0660

and provide extensive error

diagnostics. The Error Messages

mistakes. Using SSP/PC, scientists

save the user from inadvertent

and engineers can save time by

SSP/PC functions include:

and Statistical Fcns, 15 Random

 $l_{\nu}(z)$ ,  $K_{\nu}(x)$ , Ai(x), Bi(x), Ai'(x),

Number Generators, Ei(x),  $E_n(x)$ ,

and difficult programming.

freeing themselves from tedious

34 Elementary Fcns, 18 Probability

li(x), Si(x), Ci(x),  $\Gamma(x)$ ,  $\psi(x)$ ,  $B(x,\omega)$ ,

 $l_x(a,b)$ , erf x, S(x), C(x),  $J_v(z)$ ,  $Y_v(x)$ ,

Bi'(x), ber x, bei x, ker, x, kei' x,

K(x), E(x),  $F(\rho|a)$ ,  $E(\rho|a)$ ,  $\Pi(\rho|a,b)$ ,  $\Lambda(a,b,\rho)$ ,  $\mathcal{P}(z)$ ,  $\mathcal{P}'(z)$ ,

 $C^{(a)}(x)$  and many more. \$350.00

LATTICE NOW OFFERS

CODE SIFTER

light CPU users.

drudgery. \$119.95

 $P_n(x), H_n(x), L^{(a)}(x), J^{(a,\beta)}(x), G_n(p,q,x),$ 

Code Sifter is a software develop-

ment tool that enables programmers

to write faster executing software.

that indicate which code sections

information you can concentrate

your optimization efforts on the

and ignore the routines that are

are the heavy CPU users. Using this

areas that are really the bottlenecks

A major advantage of Code Sifter

over other products of this type is

architecture or assembly language. Link map listings are optional. In

have knowledge of the machine

most cases Code Sifter can set up

divide them automatically, freeing the programmer from a lot of

the ranges and repeatedly sub-

that it does not require that the user

It produces CPU usage statistics

The Model E232-51 is a new in-circuit emulator from Signum Systems that provides real-time, transparant emulation for the 8031, 8051, and 8751 microcontollers. Connected to an IBM PC via the RS-232 interface, Model E232-51 features complete debugging facilities. Along with a command-driven user interface, the emulator provides users with windowing software and mouse support for controlling and monitoring program execution. Model E232-51 with 64K of overlay program memory is priced at \$3,195. Reader Service No. 37. Signum Systems

1820 14th St., Ste. 203 Santa Monica, CA 90404 (213) 450-6096

Lang-Allan has announced Version 2.00 of Bluestreak Plus, its communication package for the IBM PC, PC/XT, PC/AT, and compatibles. Bluestreak Plus is a full-featured software package that combines PC-to-PC communication and PC-to-mainframe communication with an openarchitecture format for customization and modification at any level. The programming interface allows you to develop applications in many popular languages such as C, assembly language, Turbo Pascal, and dBASE. Bluestreak Plus 2.00 sells for \$89. Reader Service No.

Lang-Allan Inc. 2457 Aloma Ave., Ste. B Winter Park, FL 32792 (305) 677-1539

Full-function simulation models of the Motorola MC68000 and MC68010 miroprocessors are available from Quadtree along with an extensive library of models of associated peripherals and an optional graphic microprocessor development system. The new models are part of Quadtree's Designer's Choice library, a library of software simulation models of standard, off-the-shelf digital devices. Call for prices. Reader Service No. 39.

Quadtree Software Corp. 1170 Rt. 22 East Bridgewater, NJ 08807 (201) 725-2272

A 3½-hour C programming course is

## Lattice Works

#### SCREEN DESIGN AID (SDA) IS NOW AVAILABLE FOR RPG II PROGRAMMERS

The Lattice Screen Design Aid (SDA) utility helps Lattice RPG II programmers create and modify display screen formats during the development and testing of application programs. Instead of coding S and D specifications for the SFGR, SDA allows you to build displays directly on your PC. When the displays on the screen are as you want them. SDA creates the SFGR source file, the screen format file for the RPG program and the skeleton RPG program for the WORKSTN file; and it can optionally print out a source listing. This product now joins Lattice Sort/Merge (LSM™) and Source Entry Utility (SEU) in supporting the Lattice RPG II compiler. \$350.00

## LATTICE ANNOUNCES NEW SCIENTIFIC SUBROUTINE PACKAGE

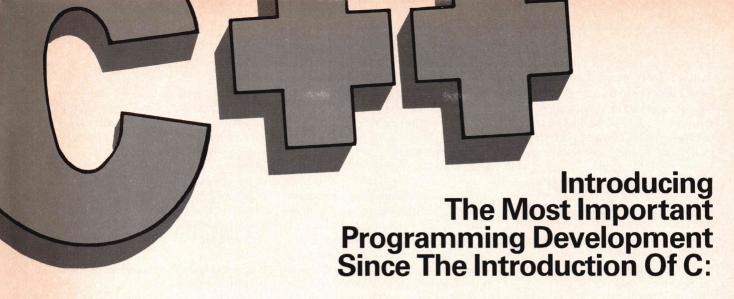
SSP/PC is a library of mathematical subroutines essential to scientific, engineering and statistical computations. Comprised of more than 145 subroutines callable from FORTRAN, Pascal, BASIC and C, SSP/PC is as extensive as similar packages generally used on mainframe computers. The routines are very fast and extremely accurate



(312)858-7950 TWX 910-291-2190

INTERNATIONAL SALES OFFICES: Benelux: Ines Datacom (32) 2-720-51-61 Japan: Lifeboat, Inc. (03)293-4711 England: Roundhill (0672)54675 France: SFL (1)46-66-11-55 Germany: Pfotenhaur (49)7841/5058 Hong Kong: Prima 85258442525 A.I. Soft Korea, Inc. (02)7836372

Circle no. 101 on reader service card.



## ADVANTAGE C++

## Exclusively From LIFEBOAT For PC/MS-DOS

Finally there's a programming language that enhances C, instead of making it obsolete!

ADVANTAGE C++, developed by AT&T, is a major programming breakthrough. By introducing the concept of classes, it enables C programmers to use object-oriented programming methods.

ADVANTAGE C++ gives you greater efficiency, flexibility and reliability than ever before—and allows you to more productively build large and sophisticated applications.

## ADVANTAGE C++ All The Benefits Of C Without Its Limitations!

- Opens the door to object-oriented programming.
- Allows programs with greater resilience and fewer bugs.
- Lets you write reliable, reusable code that is easier to understand.
- Has many enhancements over C, yet maintains full compatibility with existing C programs.

## Advantages Only ADVANTAGE C++ Can Give You:

- Operator Overloading—Allows simple, reliable user-designed types.
- Function Name Overloading—Simplifies function names and argument lists.
- Guaranteed Programmed Initialization Ensures automatic initialization of all data objects before their use.
- Guaranteed Programmed Type Conversion— Ensures consistent conversions from one userdesigned type to another.
- Optional Strong Type Checking—Weeds out type mismatches at compile time.

- Classes Similar to structures; provide syntax for user-designed data types and encapsulation of access functions with data objects.
- Data Abstraction Makes code easily reusable and more resilient.
- Data Hiding—Improves software reliability.
- Inheritance Enables generic code written for more abstract types such as 'list' or 'windows' to be used by more specific types.
- Constant Data Types—Prevent inadvertent alteration of fixed values, such as hardware addresses.
- Reference Data Types—Improve the efficiency of argument passing.
- Inline Functions—Remove the overhead of calling external functions.
- Heap Management—Simplifies the use of dynamic memory.

Why be limited to just C...when you can have all these pluses! ADVANTAGE C++ includes libraries for stream I/O and complex math—and versions are now available for Lattice C and Microsoft C. Call today to order or to obtain a complete technical specification sheet:

1-800-847-7078 In NY: 914-332-1875

## LIFEBOAT

The Full-Service Source for Programming Software.

Circle no. 359 on reader service card.

(continued from page 146)

available on video cassette from Berkeley Decision/Systems. The course is designed for expert programmers who want to learn C or people who are familiar with the language and want to learn more about it. The course includes a manual and more than 40 complete C programs to demonstrate the concepts presented in the video. A Programmer's Introduction to C is available for \$400 in VHS or Beta formats. Reader Service No. 40. Berkeley Decision/Systems

150 Belvedere Terr. Santa Cruz, CA 95062 (408) 458-0500

Zoom Telephonics is now shipping a 2,400-baud version of its Zoom/Modem PC 1200. The new version has such additional features as demon dialing, audio input and output ports, and a high-speed 16450 UART for assured compatibility with IBM PC/ATs and compatibles. The Zoom/Modem PC 1200 can be upgraded to 2,400 baud with a plug-in board that costs \$249. The Zoom/Modem PC 2400 ST sells for \$499, and an XL version with more features costs \$50 more. Reader Service No. 41.

Zoom Telephonics Inc. 207 South St.

Boston, MA 02111 (617) 423-1072

Access Associates has introduced Alegra, a memory expansion unit designed to add 512K of external memory to the Commodore Amiga. Alegra has a small footprint and allows for future expansion of up to 2 megabytes by replacing memory and configuration devices. The unit supports the auto-configuration architecture of the Amiga, and power is supplied by the computer at the expansion connector. Alegra sells for \$379. Reader Service No. 42.

**Access Associates** 491 Aldo Ave. Santa Clara, CA 95054 (408) 727-8520

Digitronix has released a low-cost Turbo upgrade kit called Veloz that

brings IBM PC/XTs and compatibles up to the speed of a PC/AT. Veloz offers 100 percent compatibility with all major software packages and can be run with either the 8088-2 or V20 with no need to power down or replace the CPU. The price is \$98. Reader Service No. 43.

Digitronix 2135 Junction Ave. Mountain View, CA 94043 (415) 964-5103

PAX from Baker & Rabinowitz is a real-time multitasking executive for the IBM PC. It runs in concert with MS-DOS and requires no licensing or incorporation fees. The system kernel supports up to 32 concurrent tasks and is fully preemptive. The package is priced at \$149.95. Reader Service No. 44.

Baker & Rabinowitz Inc. 3869 Kilbourne Ave. Cincinnati, OH 45209 (513) 871-0886

CodeWorks is a new magazine for

## Two great reasons to buy Turbo Pascal: System Builder \$9995 and Report Builder \$7500

#### From the Designer Series by Royal American Technologies.

State-Of-The-Art Program Generators that automatically build a Relational Database system without coding. Entry level "coders" can produce Database systems without coding. Developers have more flexibility and horsepower than any development tool on the market.

Self-documenting program includes screen schematics. System Builder will generate 2,000 lines of program code in approximately 6 seconds.



TASK

Planning and Design

Screen Painting

Elapsed time to

completed system

Programming

"I think it's wonderful . . . prospective buyers should seriously consider DESIGNER even before

Corporate Accounts Manager, Computerland

"We used DESIGNER last year to program a major application. It saved our programmers so much time. We now use DESIGNER instead of dBASE III as our development standard."

Mr. Peter Barge, Director Services Division, Horwath & Horwath

SYSTEM BUILDER DBASE III™

60 minutes

3 hours

10 hours

14 hours

faces to a maximum of 16 Datafiles created with System Builder • Supports Global Parameters such as Headings, Footers, Lines Per Page, Print Size and Ad Hoc Sorting • Page breaks on Sub-Totals • Reports can also in-clude Text Strings, Variables or Computed expressions containing refer-ences from up to 16 Datafiles • Use range input screens allow End Users to select portions of a report as needed (i.e. specific account ranges can be requested) • Easy-to-use Interface Program included to access dBase Files

SYSTEM BUILDER PERFORMANCE (Typical 10 screen 8 file/index application)

#### SYSTEM BUILDER FEATURES:

\*\*Automatically generates Indented, Structured, Copy Book Source Code ready for compiling with Turbo Pascal (no programming needed)

\*\*Paint Application and Menu screens using Keyboard \*\* Screens all use In-Line machine code for exceptional speed \*\* 16 Datafiles and 16 Index Keys per application \*\*Paint functions include: —Center, copy, move, delete, insert or restore a line with one keystroke —Cut and paste blocks of text screen to screen —Draw and erase boxes —Access special graphic characters and character fill —Go straight from screen to screen —Define colors and intensities \*\*Support an unlimited number of memory variables \*\* Fille Recovery Program \*\* automatically modify existing datafiles \*\* Experienced developers can modify the System Builder \*\* Develop systems for Floppy or Hard Disk \*\* Modify System Builder \*\* Develop systems for Floppy or Hard Disk \*\* Modify System Builder \*\* coupts source code to include External Procedures, Functions and Inline Code \*\* Easy-to-use Interface Program included to access ASCII and Dbase Files

#### **REPORT BUILDER FEATURES:**

• Automatically generates Indented, Structured Source Code ready for compiling Turbo Pascal (no programming needed) • Automatically inter-



VARS, System Integrators and Dealers, let's work together. Head office: (415)397-7500

60 minutes

15 minutes

17 minutes

1 hour and

**Royal American Technologies** 201 Sansome, Suite 202 San Francisco, CA 94104

(800) 654-7766 in California (800) 851-2555 Ask for Operator 102.

copies of SYSTEM Please rush me: \_ BUILDER at \$99.95 per copy; \_\_\_\_ copies of REPORT BUILDER at \$75.00 per copy I've enclosed \$5.00 for postage and handling California residents add 6% sales tax.

Address	
City	
State	Zip

Payment: Check Money Order Cashiers Check □ AMEX □ VISA □ MASTERCARD

Card Number

30-Day Money-Back Guarantee. Not copy-protected. \$10 restocking fee if envelope is opened. \$ystem Requirements: IBM PCIXTAT', or smillar, with minimum 256K, RAM, dual floppy drives, or hard disk, color or monochrome monitor, MS; or PC DOS' version 2.0 or later, Turbo Pascal Version 2.0 or later (Normal, BCD or 80B', Persions).

people interested in BASIC programming under MS-DOS, TRS-DOS, or CP/M. Each annual subscription covers all six issues of the calendar year and costs \$24.95. Interested readers can write for a free sample issue. Reader Service No. 45.

CodeWorks Sample Copy Offer 3838 South Warner Tacoma, WA 98409

A system modeling tool called Performance Analysis Tool Box is available from Computer Technology Associates. The package simulates a variety of centralized or distributed computer architectures, allowing designers to investigate broad ranges of use patterns in a hypothetical system. It's \$10,000 for the IBM PC. Reader Service No. 46.

Computer Technology Associates 7927 Jones Branch Dr., #600W McLean, VA 22102

Microsoft Corp. has announced the availability of extensions to the MS-DOS operating system that support the use of CD ROM disk drives with personal computers. The MS-DOS CD ROM extension consists of two software modules-a hardware-independent program and an installable device driver that must be customized by each manufacturer to work with its own hardware. Microsoft will supply the hardware-independent program—the DOS extension that will handle the much higher capacity of the CD ROMs—plus a sample device driver and documentation.

With the new software, PCs running DOS 3.1 or 3.2 can read data from any CD ROM disk that is compatible with the High Sierra Group file format proposed at the National Computer Conference in May 1986. Microsoft will license these extensions directly to CD ROM drive manufacturers, and they are available only on an OEM basis. Reader Service No. 47.

Microsoft Corp. 16011 N.E. 36th Way P.O. Box 97017 Redmond, WA 98073-9717 (206) 882-8080

**Practical Peripherals** is now offering a stand-alone 1,200-bps modem,

the Practical Modem 1200 SA. It is fully Hayes-compatible, includes auto-dial/auto-answer capabilities, supports virtually all communications software, and includes an upgrade path for a programmable enhancement card. The suggested retail price of the modem is \$239. Reader Service No. 48.

Practical Peripherals 31245 La Baya Dr. Westlake Village, CA 91362 (818) 991-8200

DogStar Software has announced subVol, a disk subvolume manager that brings PC-DOS- and ProDOS-like subdirectories to Apple Pascal DOS. SubVol works on any Pascal formatted disk device and allows hard-disk users to format directly and install a complete set of subvolumes with Apple Pascal.

The program works by attaching virtual disk drivers to the unused disk units in an Apple Pascal system. The virtual disk drivers cause a portion of

any real disk to behave as though it were a volume in itself, including a subdirectory plus any files in that subvolume. The product runs on Apple II computers with Apple Pascal (Version 1.1, 1.2, or 1.3). The price is \$34; with source code it costs \$75. Reader Service No. 49.

dogStar Software P.O. Box 302 Bloomington, IN 47402 (812) 333-5616

Instant Replay by **Nostradamus** is a demonstration development toolkit that generates tutorials, demos, presentations, menu systems, and timed keyboard macros. It is not copy-protected and runs on the IBM PC, PC/XT, and PC/AT. The product is priced at \$89.95. Reader Service No. 50.

Nostradamus 5320 South 900 E, Ste. 110 Salt Lake City, UT 84117 (801) 261-0769

DDJ



# AT LAST, A HIGH-QUALITY, EASY-TO-USE C COMPILER WITHOUT THE HIGH PRICE



The Eco-C88 C compiler has everything you need for professional program development, including: a standard library of over 200 functions, cc and "mini-make" utilities, ANSI language enhancements (e.g., prototyping, enum, void), and user's manual. For a limited time, we'll even give you a full-screen program editor. Here's what some reviewers are saying about the Eco-C88 C Compiler:

"Eco-C is the first compiler reviewed that has clearly begun implementing the ANSI standard... Eco-C performed well on all the benchmarks, generating code that was quite comparable to that of compilers 10 times as costly."

Christopher Skelly, Computer Language, Feb., 1986

"The (cc) driver program is another strength... This compiler does handle syntax errors much better than average – no avalanche of spurious messages here."

William Hunt, PC Tech Journal, Jan., 1986

"Eco-C88 is a high-quality package ... one of the fastest ... convenient to use ..."

Dr. David Clark, **Byte**, Jan., 1986

"Eco-C is definitely a bargain . . . it includes both the compiler and an excellent Turbo-style editor." Gary Entsminger, **Micro Cornucopia**, April-May, 1986

The complete compiler package has everything you need, all for the low price of only \$59.95. Also ask about our support products, too!

1-800-952-0472 (orders) 1-317-255-6476 (info)



Ecosoft Inc. 6413 N. College Ave. Indianapolis, IN 46220

**ECOSOFT** 

# OGITECH MODULA-2/80

## \$89 Price

- Separate Compilation
- Native Code Generation
- Large Memory Model Support
- Multitasking
- Powerful Debugging Tools
- Comprehensive Module Library
- Available for the PC and the VAX Use LOGITECH MODULA-2/86 to decrease your overall development cycle and produce more reliable, more maintainable code.



#### LOGITECH MODULA-2/86

\$89

Includes Editor, Run Time System, Linker, 8087 Software Emulation, Binary Coded Decimal (BCD) Module, Logitech's comprehensive library, Utility to generate standard .EXE files. AND more!



#### **LOGITECH MODULA-2/86** with 8087 Support \$129



#### **LOGITECH MODULA-2/86** PLUS \$189

For machines with 512K of RAM. Increases compilation speed by 50%.



#### RUN TIME DEBUGGER (Source level!)

The ultimate professional's tool! Display source, data, call chain and raw memory. Set break points, variables, pinpoint bugs in your source!



#### **UTILITIES PACKAGE \$49**

Features a Post-Mortem Debugger (PMD). If your program crashes at run-time the PMD allows you to analyze the status of the program and locate the error. Also includes a Disassembler, Cross Reference Utility, and Version that allows conditional compilation.



#### LIBRARY SOURCES

899

Source code now available for customization or exemplification.



#### WINDOW PACKAGE

\$49

Build windows into your programs. Features virtual screens, color support, overlapping windows and a variety of borders.



#### **MAKE UTILITY**

\$29

Figures out dependencies and automatically selects modules affected by code changes to minimize recompilation and relinking.

#### **CROSS RUN TIME Debugger and ROM Package**

Still available at an introductory price!

#### TURBO PASCAL to \$49 **MODULA-2 TRANSLATOR**

'Turbo Pascal... is a very good system. But don't make the mistake of trying to use it for large programs?

Niklaus Wirth\*

Our Translator makes it even easier for Turbo users to step up to Modula-2/86. It changes your Turbo source code into Modula-2/86 source, solves all the incompatibilities, and translates the function calls of Turbo into Modula-2/86 procedures. Implements the complete Turbo libraries!

Call for information about our VAX/VMS version, Site License, University Discounts, Dealer & Distributor pricing.

30 Day Money Back Guarantee! To place an order call our special toll free number:

800-231-7717 in California

800-552-8885

Special Holiday Offer

Step up to the power of LOGITECH MODULA-2/86 at a saving of nearly \$100 off our usual low prices! We're offering a complete tool set including our compiler with 8087 support (for use with or without an 8087), our Turbo to Modula-2/86 Translator, Run Time Debugger, and Utilities in one holiday package at a special price!

### YES I want to step up to LOGITECH MODULA-2/86!

Here's the configuration I'd like:

☐ Special Holiday Package \$199 ☐ Logitech Modula-2/86 \$89 ☐ with 8087 support \$129 □ Plus Package \$189 ☐ Turbo to Modula Translator \$49 ☐ Run Time Debugger \$69 \$49 ☐ Utilities Package ☐ Library Sources 899 \$49 ☐ Window Package ☐ Make Utility \$29 ☐ ROM Package \$199

Add \$6.50 for shipping and handling, Calif. residents add applicable sales tax. Prices valid in U.S. only.

Total Enclosed

☐ Visa ☐ MasterCard ☐ Check Enclosed

Card Number

Expiration Date

Signature

Name

Address

City

Zip

Phone



State

Logitech, Inc. 805 Veterans Blvd. Redwood City, CA 94063 Tel: 415-365-9852

In Europe: Logitech SA, Switzerland Tel: 41-21-879656

In Italy: Tel: 39-2-215-5622

\*as reported in Micro Cornucopia. August-September 1985, Turbo Pascal is a registered trademark of Borland International.

#### ADVERTISER INDEX

Read	er	Reade	er
Servi	ce Page	Service	
No.	Advertiser No.	No.	Advertiser No.
92	Addison Wesley	286	Microcompatibles 142
350	Aldebaran Laboratories	*	Micromint
219	Allen Emerson & Franklin 89	154	Microport Systems, Inc
321	Alpha Computer Service	105	Microprocessors Unlimited 102 Microsoft
360 258	Austin Code Works	156	Mix Software
115	Barrington Systems, Inc4-5	249	Mortice Kern Systems, Inc 131
182	BC Associates	309	Microsoft Associates72
267	Beacon Street Software, Inc 86	220	Nantucket Corporation 84
202	Berkeley Decision/Systems 91	331	Nirvonics
159	Blaise Computing 13	243	Norton Utilities (The)116
217	Blaise Computing 79	251	Nostradamus2
263	Block Island Tech 90	227	Oakland Group, Inc 47
161	Borland International	342	Oakland Group, Inc
212	Burton Systems Software	254	Oasys
235 181	C Software Toolset	357 214	Oregon Software         C3           Periscope Co. Inc.         74
*	C Ware	343	PharLap
307	Cauzin Systems	239	PMI
*	Cauzin Systems	229	Port - A - Soft
343	Circuit Research Corporation 145	129	Programmers Connection 75
122	Compu View	129	Programmers Connection76-77
237	Compuserve 139	129	Programmers Connection 71
348	Creative Computer Software 83	334	Programmer's Paradise 33
*	Creative Programming	174	Programmer's Shop
*	CSSL, Inc	133	Programmer's Shop50-51
268	Custom Software Systems	301-30 337	Programmer's Shop109
203	Datalight9	355	Quelo
353	Davidge Corporation	206	Raima Corporation
258	Desktop A.I	145	Rational Systems 55
89	Ecosoft, Inc	312	Royal American Technologies 148
90	Edward K. Rose 87	*	SAS Institute 46
173	Entelekon 67	168	Sapiens Software 82
93	Fair-Com	210	Scientific Endeavors
340	Farbware 91	85	Semi-Disk Systems
311	Gimpel Software	78 345	SLR Systems       44         Soft Cap, Inc.       90
291	Gold Hill Computers, Inc	113	Softcraft Inc
97	Greenleaf Software	259	Softfocus
351	Guidelines Software 48	361	Software Factory114
132	Harvard Softworks 53	314	Software Garden Inc 59
233	Hawaiin Village Computers 96	347	Software Masters
780	Hersey Micro Consulting 58	170	Software Security, Inc 20
293	IMSI	148	Solution Systems
327 294	Integral Quality, Inc	142 152	Solution Systems
275	Kydor Computer Systems 93	354	Solution Systems
285	Laboratory Microsystems, Inc 95	287	Stonybrook Software 80
266	Language Processors, Inc 138	345	T.O.C Business Solutions 102
181	Lattics, Inc	175	Tom Rettig Association
346	Levco127	344	True Basic142
118	Lifeboat	230	TSF 81
359	Lifeboat147	207	Turbo Power Software 56
257	Logitech, Inc	119	Turbo Tech Report
135	Lugaru	332	Unify Corporation
336 108	Magus Inc.         135           Manx Software Systems         7	316 157	Upland Software
317	Marshall Language Systems 45	112	Wendin
285	MDS, Inc	116	Wizard Systems106
352	Metamax	244	Workman & Associates 142
358	MetaCom Co	225	Xenosoft
300	Micro Way 62		

\*This advertiser prefers to be contacted directly: see ad for phone number.

#### ADVERTISING SALES OFFICES

Midwest

Michele Beaty (317) 875-8093

Southeast

Gary George (404) 897-1923

Northeast

Cynthia Zuck (718) 499-9333

Northern California/Northwest Lisa Boudreau (415) 366-3600

Southern California/AZ/NM/TX Michael Wiener (415) 366-3600

**Advertising Director** 

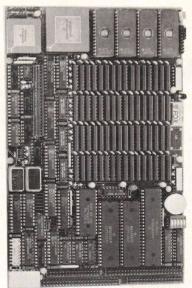
Robert Horton (415) 366-3600

## GMX° Micro-20

Single-Board Computer

Mainframe CPU Performance on a 5.75" x 8.8" Board

(benchmark results available on request)



\$256500 12.5 MHz Version Quantity Discounts Available

#### **Features**

- 32-Bit MC68020 Processor (12.5, 16.67, or 20MHZ)
- MC68881 Floating-point coprocessor (optional)
- 2 Megabytes of 32-bit wide, high-speed RAM
- 4 RS-232 Serial I/O Ports (expandable to 36)
- 8-bit Parallel I/O Port ('Centronics' compatible)
- . Time-of-Day Clock w/battery backup
- 16-bit I/O Expansion Bus
- Up to 256 Kbytes of 32-bit wide EPROM
- Floppy Disk Controller for two 5¼" drives
- SASI Intelligent Peripheral Interface (SCSI subset)
- . Mounts directly on a 514" Disk Drive
- Optional Boards Include Arcnet, Prototyping, I/O Bus Adapter, 60 line Parallel I/O, RS-422/485

#### Software

#### Included:

- GMX Version of Motorola's 020Bug Debugger with up/download, breakpoint, trace, singlestep, and assembler/disassembler capabilities
- · Comprehensive Hardware Diagnostics

#### Optional:

UNIX™-like Multi-user/Multi-tasking Disk Operating Systems

- 0S-9/68000™ (Real-time and PROMable)
- UniFLEX™

Programming Languages and Application Software

- · BASIC, C, PASCAL, ABSOFT FORTRAN, COBOL and ASSEMBLER
- Spreadsheet, Data Base Management, and

COMPLETE EVALUATION SYSTEMS AVAILABLE

#### GMX Inc. 1337 West 37th Place Chicago, IL 60609

(312) 927-5510 • TWX 910-221-4055 State-of-the-Art Computers Since 1975

Circle no. 311 on reader service card.

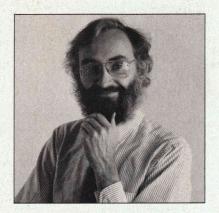
### SWAINE'S FLAMES

7 es, I was kidding about BASIC in November. BASIC, on the other hand, seems to be getting serious, with such features as advanced structure constructs, new data types, labels for branching, internal and external subroutines, multiline and decision-making functions, recursion, libraries, modules, better file I/O, low-level and DOS access, sophisticated string manipulation, and advances in portability and standardization. And now that Borland is bringing out a Turbo BASIC, competition may hasten the maturation process.

Granted that the Turbo Pascal-Turbo BASIC-QuickBASIC type of rapid-interaction compiler is not a reasonable substitute for C and assembly language in a major software development task, it does seem to be finding a place in certain kinds and stages of development. The fact that both Borland and Microsoft have interactive C compilers in the works suggests that they think so-it's hard to believe that they expect to sell C to Sundayafternoon programmers. But it's more significant that there seems to be a growing interest in enlarging the programmer's toolkit, offering more options in development environments, and offering tools that make the programmer more productive and efficient.

Good programming is often artful. Any engineering discipline is like art in that both bring new things into the world and rely on skill and serendipity and unlike art in being more concerned with result than with process. In fact, one goal of engineering is to refine processes, automating portions in order to free the engineer to be artful at another level. Does software development do this?

If not, it should. I recently reread Robert Heinlein's *The Door into Summer* and was struck by how convinc-



ingly and appealingly Heinlein portrayed the engineer as an artist whose very art provides the means to improve his brush. If software development really is an engineering discipline, it needs more Waldoes and Drafting Dans, more tools to automate the repetitive processes.

John Backus does not believe that programming is an engineering discipline yet, arguing that it is still too much an art. (See "From Function Level Semantics to Program Transformation and Optimization," *Lecture Notes in Computer Science* 185 [1982].) Solving the software crisis, he maintains, requires that it become an engineering discipline.

Parallel processing is the wave of the future, right?

That was the gist of Gordon Bell's keynote address at the Fall Joint Computer Conference in Dallas in November. The promise of parallelism may have been overstated in some areas. Two former advocates of parallel design for database machines, Haran Boral of the Israel Institute of Technology and David DeWitt of the University of Wisconsin at Madison, have come to believe that we should not build database machines that attempt to maximize throughput via massive parallelism. The problem, they contend, is that I/O bandwidth per gigabyte of storage is decreasing rapidly and that it is the I/O bandwidth issue that will be the bottleneck in database machines. "Unless mechanisms for increasing the bandwidth of mass storage devices are found, highly parallel database machines are doomed to extinction," they conclude in "Database Machines: An Idea Whose Time has Passed?" (Database Machines, H. O. Leilich and M. Missikoff, eds. [Berlin: Springer-Verlag, 1983]).

Those of you who live in California may have discerned a bit of chauvinism in Massachusetts' claiming to be the Software Capital of the country. Those of you who do not live in California probably detected even more chauvinism in the recent passage of a ballot proposition making English California's state language. No doubt the California legislature will soon be turning arroyos and mesas into gulches and hills just as the French, who invented chauvinism, have for years been kicking Americanisms out of la langue Française.

My cousin Corbett is cursing what he calls "egotistical linguistic chauvinism." He had worked hard on an alternative proposition that he thinks is much more appropriate because it actually addresses a real problem. Unfortunately, it did not get on the ballot in 1986, but Corbett plans to be more successful in 1988 and is starting to organize now. His plan is to replace English with C as California's language, and he invites your participation. C, he points out, is capable of expressing anything anyone could ever need to express, and the time has come for fanatical supporters of minority languages to put aside their pride, accept the inevitable, and end the Babel of incommensurable dialects once and for all.

"Des egos, et encore des egos."—Ed Faber, in Silicon Valley, the French edition of Fire in the Valley by Paul Freiberger and me.

> Michael Swaine Michael Swaine editor-in-chief



## TALK OF THE TOWN

One language supports this community.

That language is Pascal-2, now on the PC and producing the fastest, most compact code available. For the professional programmer, imagine what you can do with this power.

you can do with this power:

Cut execution time by 20% to 200%

Transport MS-DOS programs to VAX,
PDP-11, and 68000 machines with only
minor adjustments Cut executable
program size by up to 50% Use all of
DOS-addressable memory through
efficient large-memory model Speed
error correction and save development
turn-around time with sophisticated error
checking and reporting Find and fix
logical errors with the interactive sourcelevel debugger Access DOS services

## Pascal-2°

FOR MS-DOS



and network files ■ Call Microsoft
FORTRAN, C, Pascal, and assembler
■ Upgrade from TURBO Pascal with
compatible strings, equivalent procedures
and access to TURBO graphics.

Plus!

■ Intel CEL87 mathematical library for scientific computing ■ A special interface between Pascal-2 and the programmable BRIEF text editor (editor optional). ■ Certified ISO standard Level 1.

Dramatically improve your productivity and introduce your PC software to the VAX next door.
Call or write OREGON SOFTWARE, INC.

Call or write OREGON SOFTWARE, INC. 6915 SW Macadam Avenue, Portland, OR 97219 (800) 367-2202 TWX: 910-464-4779 FAX: (503) 245-8449

OREGON SOFTWARE

Real tools for real work

#### AT LAST THE PERFORMANCE IS PORTABLE

The following are trademarks: Oregon Software, Pascal-2, Oregon Software, Inc.; IBM, PC-AT, PC-DOS International Business Machines Corporation; Intel, Intel Corporation; MS, Microsoft Corp.; TURBO Pascal, Borland International, Inc.; BRIEF, UnderWare Corp.; PDP, VAX, Digital Equipment Corp.

Circle no. 357 on reader service card.

## Why more than half a million people are using Turbo Pascal

Because Turbo Pascal is faster than any other Pascal compiler—it's become the industry standard. And because Turbo Pascal is backed by a complete range of "tool boxes" that give you most of the programming tools you'll ever need.

> The Turbo Pascal family is never static, but is continuously expanding, with new products like Turbo Editor Toolbox and Turbo Gameworks.

Language deal of the century...Turbo Pascal Jeff Duntemann, PC Magazine Turbo Pascal has got to be the best value in languages on the Jerry Pournelle, BYTE Magazine market today This compiler, produced by Borland International, is one of the best programming tools presently available for the PC Michael Covington, PC Tech Journal 33

The secret of software success is not merely low price, but top quality allied with complete documentation, like our 400-page reference manual.

All of which are some of the reasons why Turbo Pascal is clearly the leader, and the recipient of awards like PC Week's "Product of the Year" and PC Magazine's "Award for Technical Excellence."

#### TNT\* A DYNAMITE DEAL!

\*Turbo 'N Tutor Turbo Pascal and Turbo Tutor for only \$125.00. You save nearly \$15.00, and have everything you need to get up and go with Turbo Pascall



- Turbo Pascal® 3.0 with 8087 and BCD
- Turbo Database Toolbox"
- Turbo Graphix Toolbox®
- Turbo Tutor® 2.0
- Turbo Editor Toolbox™
- Turbo GameWorks®

#### A whole family of tools

Success breeds success, so the Turbo Pascal family has flourished. Your choices now include:

☐ Turbo Pascal 3.0 combines the fastest Pascal compiler with an integrated development environment, and 8087 math co-processor support and Binary Coded Decimals.

☐ Turbo Database Toolbox a perfect complement to Turbo Pascal. It includes a complete library of Pascal procedures that allows you to search and sort data and build powerful database applications.

☐ Turbo Graphix Toolbox includes a library of graphics routines for Turbo Pascal programs. Lets even beginning programmers create highresolution graphics with an IBM,®

Amazing value! Turbo Editor Toolbox includes MicroStar, a full-blown editor that also does windows! You get ready-to-compile source code and a 200-page manual that tells you how to integrate the editor procedures and functions into your programs.

Turbo GameWorks gives you the games you can play, write, rewrite, bend and amend! Turbo GameWorks reveals the secrets and the strategies of game design. You're given source code, a 200-page manual, and the insight

needed to write and customize your own irresistible games

to build your own word processor!

Also includes ready-to-play Chess, Bridge, and Go-Moku-an ancient Japanese game that can divert you from reality for hours on end.

Hercules,™ or compatible graphics adapter. Does complex business graphics, easy windowing, and stores screen images to

☐ Turbo Tutor 2.0 teaches you step-by-step how to use Turbo Pascal, with commented source code for all program examples on diskette.

#### An offer you can't refuse

You can save \$119.75 when you choose the Turbo Jumbo Pack. Six different Turbo Pascal products for only \$299.95.

You get Turbo Pascal 3.0 and Turbo Editor Toolbox and Turbo Tutor 2.0 and Turbo Graphix Toolbox and Turbo GameWorks and Turbo Database Toolbox!

So act now-rush to your dealer or call us today!

Also includes MicroStar, a complete

editor with full windowing capabilities.

(You could pay \$100.00 or more for a

program like MicroStar, but you get it

free as part of our Turbo Editor Toolbox.)

You can also use Turbo Editor (which of

course integrates with Turbo Lightning®)

ES! I warn the best! To order by phone, or for a dealer nearest you, call (800) 255-8008 in CA call (800) 742-1133 Turbo Pascal 3.0 w/8087 & BCD \*\* Turbo Pascal for CP/M-80 69.95 \$ Turbo Database Toolbox Turbo Graphix Toolbox Turbo Editor Toolbox Turbo GameWorks\* Turbo Pascal & Turbo Tutor Turbo Jumbo Pack\* Outside USA add \$10 per copy CA and MA res. add sales tax Prices include shipping to all US cities Carefully describe your computer system Mine is: \_\_ 8-bit \_\_ 16-bit l use: \_\_ PC-DOS \_\_ MS-DOS \_\_ CP/M-80 \_\_ CP/M-86 My computer's name and model is: The disk size I use is: □ 3½\* □ 5¼\* □ 8\* VISA MC Shipping Address: CODs and purchase orders WILL NOT be accepted by Borland. Outside USA make payment by bank draft, payable in US dollars drawn on a US bank NOT COPY PROTECTED 60-DAY MONEY-RACK GUARANTEE If within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department and we will gladly arrange a refund. All prices are suggested list prices and are subject to change linimum System Requirements: Turbo GameWorks, Turbo Graphix Toolbox, Turbo Tutor, & Turbo Editor Toolbox—192K. All other products, 128K. \*IBM PC, PCjr, AT, XT, and true compatibles.

4585 SCOTTS VALLEY DRIVE SCOTTS VALLEY, CA 95066 (408) 438-8400 TELEX: 172373

Borland products include Turbo Prolog; Turbo Pascal; Turbo Pascal for the Mac; Turbo Tutor; Turbo Editor Toolbox; Turbo Database Toolbox; Turbo Graphix Toolbox; Turbo GameWorks; Turbo Lightning, Lightning Word Wizard; Rellex The Analyst; Reflex for the Mac; Reflex Workshop; SideKick; SideKick; and SuperKey—all of which are trademarks or registered trademarks of Borland International, Inc. or Borland/Analytica, Inc. Traveling SideKick is not in any way associated with Traveling Software, Inc. of Seattle, Washington.

CP/M-80 is a registered trademark of Digital Research, Inc. MS-DOS is a registered trademark of Microsoft Corp. IBM is a registered trademark of International Business Machines Corp. Hercules is a trademark of Hercules Computer Technology.

Circle no. 161 on reader service card.

